DOCUMENT RESUME

ED 324 215 SE 051 594

AUTHOR Lyons, Jim; Lamarre, Colin
TITLE Challenge '89: Interfacing of Chemical Instruments to Computers.
INSTITUTION John Abbott Coll., Sainte Anne de Bellevue (Quebec).
PUB DATE Aug 89
NOTE 96p.; For the 1988 version, see SE 051 593. Contains some broken type which may not reproduce well.
PUB TYPE Computer Programs (101)

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Chemical Analysis; Chemistry; *Computer Software; Computer Uses in Education; Foreign Countries; Instrumentation; Laboratory Procedures; Microcomputers; Science Activities; Science Instruction; *Spectroscopy; Visual Impairments
IDENTIFIERS *Laboratory Interfacing; PASCAL Programing Language

ABSTRACT

This project involved interfacing of microcomputers with three chemical instruments--Nuclear Magnetic Resonance (NMR), Infrared Spectroscopy (IR), and the spectrophotometer. A Pascal program called "Spectrum" allows data from the NMR to be read and graphed, a specific area of the graph zoomed, ratios of specified areas of the graph measured, the range between points in hertz determined, the averaging out of a line, comparing and crossing of different graphs, moving a graph, and saving and retrieval of graphs in three different methods. The sweep of the NMR is also controlled by the analog/digital converter and the program when receiving data from the instrument. Real-time plots by controlling the sweep from the keyboard are also possible. The Pascal program written for the IR allows data from the IR to be read and graphed, specific areas of the graph to be enlarged, absorbance to be plotted, up to three graphs to be viewed at the same time, and saving and retrieval of graphs using three different methods. Voice synthesis was implemented with the spectrophotometer to aid the blind in controlling this instrument. The Pascal program can read and plot the data while reading the mode, wavelength, and data out loud at each increment. It can also read constantly at timed intervals for a blind person to make adjustments. Appendices illustrate circuit diagrams and connections between the interfaced instruments. (KR)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED324215

CHALLENGE '89:
INTERFACING OF CHEMICAL INSTRUMENTS TO COMPUTERS

JIM LYONS
COLIN LAMARRE
DEPARTMENT OF CHEMISTRY

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY
James M. Lyons

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.
Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

CEGEP
JOHN ABBOTT
COLLEGE
P.O. BOX / C.P. 2000,
STE. ANNE DE BELLEVUE, QUEBEC H9X 3L9

SE 051 594

Dépôt Légal - 1er trimestre 1990

Bibliothèque Nationale du Québec

© Copyright John Abbott College

CHALLENGE '89

INTERFACING OF CHEMICAL INSTRUMENTS TO COMPUTERS

Jim Lyons
Colin Lamarre

August 1989

JOHN ABBOTT COLLEGE
Ste. Anne de Bellevue, Quebec

The Challenge

This project involved three chemical instruments and voice synthesis. We started with the NMR to produce a program called Spectrum. It allows you to read and graph the data from the NMR, zoom a specific area of the graph, find ratios of specified areas of the graph, find the range between points in Hz, average out a line, compare and cross different graphs, move the graph and save and retrieve the graphs in three different methods. The sweep of the NMR is also controlled by the A/D converter and the program when receiving data from the instrument. You can also do a real-time plot controlling the sweep from the keyboard.

The IR was the next instrument. The program lets you read and plot data from the IR, zoom specfic areas of the graph, plot the absorbance, view up to three graphs at the same time, save and retreive the graphs in three different methods. When there is more than one graph displayed the zoom and aborbance functions will work on all of them, while the save functions will only store the original data.

For the spectrophotometer we implemented voice synthesis to perhaps aid the blind in controlling this instrument. The program can read and plot the data while reading the mode, wavelength and data out loud at each increment. It can also read constantly at timed intervals for a blind person to make adjustments.

I would like to thank Jim Lyons (my boss) for all he has done, including teaching me the basics of all the instruments, configuring the hardware, setting the guidelines for my programs and giving me the experience I will need in the future. I would also like to thank the Government for supporting the Challenge.

NMR

NMR Interfacing

The Varian EM-360 NMR spectrophotometer was interfaced to the BEST AT cmputer using the Real Time Devices Inc. ADA300 card. Initially the interface was used only to collect data via the analog to digital converter (ADC) on the ADA300. Synchronization of the EM-360 sweep and the sampling sequence was achieved by controlling the duration of the sampling and by starting it at the same time as the NMR sweep. In a later addition to the program the EM-360 sweep was controlled by the computer via a 12-bit digital to analog converter (DAC) which was wired onto the ADA300 prototyping board. A 12-bit converter was used rather than the 8-bit DAC on the ADA300 to achieve adequate resolution on the NMR sweep. The 12-bit DAC, a DAC1230 by National Semiconductor Corporation, was controlled via the parallel ports on the ADA300 board. Full technical details are given in Appendix 1.

A program, Spectrum, was written in Turbo Pascal to collect data from the NMR and display it graphically.

The program uses ports 200-208 hex, which are assigned to the A/D converter, to collect data into an array of 7190 integers (procedure getdata). It then plots every 10 of these values because 719 is the maximum width of the Hercules screen.

For zooming we decrement this ratio with a function such as:

r = 10 * (rightmark - leftmark) / 700

The index is incremented by one and multiplied by r, added to leftmark and plotted until the index reaches the limit.

Controlling the sweep is done by assigning values to ports 210, 212 and 213 hex, which are assigned to the A/D converter which is connected to the NMR. This is discussed in more detail in the appendix.

SPECTRUM

This program is designed to graph data produced by the Nmr or other instruments with the use of the A to D converter.

Graphics Adapters supported: Hercules, CGA.

Run time commands for Spectrum:

From the main menu use the arrow keys and press return to select your choice.

A/D Converter is to obtain the data directly from the Nmr. Be sure to specify the correct port number.

Load Spectrum loads the data that was saved with Save Spectrum and plots the graph.

Save Spectrum saves the data, the title and the date to a file.

Load Ascii loads an ascii file containing integers to be plotted.

Save Ascii saves the data to an ascii file.

Review Graph lets you view the graph that is currently in memory.

OS shell lets you exit to DOS to perform some tasks.

Quit exits the program.

Processing the graph.

Zoom: In the graph area, press 'Z' to zoom a specified area of the graph:

left shift + '<' moves left side left.
left shift + '>' moves left side right.
right shift + '<' moves right side left.
right shift + '>' moves right side right.
ctrl + arrow keys moves entire box.
return to select.

From whithin a zoom you can save the screen (F2), view a screen (F3), get ratios (I) and get coupling constants (C).

Ratios : Press 'I' to obtain ratios: This will integrate the sections that you mark off and give the ratios of those sections.

Use arrow keys to move arrow.
Press space to mark a spot.
Return to evaluate.
ESC to cancel.

Coupling Constants : Press 'C' to get coupling constants: This will find the range in Hz between the points you specify.

Use arrow keys to move the line.
Press the space bar to mark a spot.
Press return to evaluate.
ESC cancels the procedure.

Smooth line: Press 'A' to average out a line : This is used to smooth out a line.

Merge : Press 'M' to plot another Spectrum file along with the current display in order to compare.

Cross : Press 'X' to join two lines together by taking their average.

Find peek : Press 'F' to for real-time display of the Nmr.
Use the arrow keys to control the sweep,
Press ctrl + arrow keys for larger increments.
Return will pause the sweep.

Moving the graph : Use the arrow keys to move the graph.
Home Key to right justify the graph.
End Key to left justify.

Screen save: F2 to save the screen for quick reviewing.

Screen restore: F3 to restore the screen that was saved with F2.

Exit: ESC to exit to main menu.

```
                                        Return to evaluate.
                                        ESC to cancel.

Coupling
Constants :   Press 'C' to get coupling constants: This will find the range
                                                   in Hz between the points
                                                   you specify.

                                                   Use arrow keys to move
                                                   the line.
                                                   Press the space bar to
                                                   mark a spot.
                                                   Press return to evaluate.
                                                   ESC cancels the procedure.

mooth line:   Press 'A' to average out a line : This is used to smooth out
                                                a line.

    Merge :   Press 'M' to plot another Spectrum file along with the current
                        display in order to compare.

    Cross :   Press 'X' to join two lines together by taking their average.


ving the graph :   Use the arrow keys to move the graph.
                   Home Key to right justify the graph.
                   End Key to left justify.

Screen save:   F2 to save the screen for quick reviewing.

een restore:   F3 to restore the screen that was saved with F2.

       Exit:   ESC to exit to main menu.




                                        Colin Lamarre
                                        John Abbott College
                                        June 7, 1989.


*************************************************************************}


 64512,0,196704}     { memory sizes }

s crt, dos, graph, drivers, fonts;    { TP 5.0 units needed }

st
  uparrow=#200;           { Define keys }
  downarrow=#208;
  leftarrow=#203;
  rightarrow=#205;
  homekey=#199;
```

```
      endkey=#207;
      ctrlleft=#243;
      ctrlright=#244;
      altM=#178;
      spacebar=#32;
      esc=#27;
      return=#13;
      F1=#187;
      F2=#188;
      F3=#189;
      board = $200;       { Port address }
      null=#0;
      limit=7190;         { array size and amount of data gathered }

type chtype=(ADconvert,loadspectrum,savespectrum,loadasc,saveasc,savescreen,rest
orescreen,review,OSshell,quit);
      artype= array[0..limit] of integer;   { array to store digital values }
      st=array[0..20] of integer;
      daterec=record
                year,month,date,dayofweek:word;
              end;
      legrec=record                  { record to hold legend }
               title: string;
               date : daterec;
             end;
      filerec=record              { used for file I/O }
                plot:artype;
                legend:legrec;
              end;

var   graph_driver  :  integer;     { to initialize graph driver & mode }
      graph_mode    :  integer;
      choice        :  chtype;      { to store user's choice }
      plot : artype;                { digital values to be plotted }
      time : word;                  { length of sample }
      spot : integer;               { on x axis }
      Xmax, Ymax : integer;         { max vals for screen type }
      legend : legrec;
      filein, fileout : file of filerec;    { file I/O }
      inascii,outascii : text;
      error : boolean;
      sbar, arrow, aline, blank : pointer;  { to store images }
      xratio, yratio : integer;             { to move, expand graph }
      xaxis:integer;


procedure initialize;

(*   Links all graph drivers and fonts into EXE file to
     produce a stand alone program.                        *)

begin
  if registerbgidriver(@cgadriverproc) < 0 then writeln('error');
  if registerbgidriver(@hercdriverproc) < 0 then writeln('error');
  if registerbgifont(@triplexfontproc) < 0 then writeln('error');
  if registerbgifont(@smallfontproc) < 0 then writeln('error');
  if registerbgifont(@gothicfontproc) < 0 then writeln('error');
  detectgraph(graph_driver,graph_mode);
  initgraph(graph_driver, graph_mode, '');
  xmax:=getmaxx; ymax:=getmaxy;
  checkbreak:=false;
end;
```

```
procedure initherc;

(* Gets all images used by Hercules adapter.      *)

var size:integer;
begin
  bar(200,100,320,110);
  size:=imagesize(200,100,320,110);
  getmem(sbar,size);
  getimage(200,100,320,110,sbar^);
  cleardevice;
  moveto(100,25);
  lineto(115,21);
  lineto(103,14);
  lineto(100,25);
  floodfill(105,20,1);
  setlinestyle(0,0,thickwidth);
  line(106,23,114,37);
  line(108,22,116,36);
  size:=imagesize(100,14,116,37);
  getmem(arrow,size);
  getimage(100,14,117,37,arrow^);
  setlinestyle(0,0,normwidth);
  cleardevice;
  line(35,0,35,315);
  size:=imagesize(34,0,36,315);
  getmem(aline,size);
  getimage(34,0,36,315,aline^);
  cleardevice;
  size:=imagesize(65,40,330,150);
  getmem(blank,size);
  getimage(65,40,330,150,blank^);
end;


function GetKey:char;

(*  Returns key value. *)

var
  C : Char;
begin
  C := ReadKey;
  repeat
    if C = NULL then
    begin
      C := ReadKey;
      if Ord(C) > 127 then
        C := NULL
      else
        GetKey := Chr(Ord(C) + 128);
    end
    else
      GetKey := C;
  until C <> NULL;
end; { GetKey }
```



```
function getinput(message:string):string;

(* Lets user type input in graphics mode. It saves the screen section
   then replaces it when done.                                        *)
```

```
var ch : char;
    cursor,csize,y : integer;
    t : string;
    replace:pointer;

begin
  if graph_driver = hercmono then
  begin
    getmem(replace,imagesize(65,40,330,150));
    getimage(65,40,330,150,replace^);
    putimage(65,40,blank^,normalput);
    settextstyle(smallfont,0,7);
    csize := textwidth('A');
    t:='';
    cursor:=80; y:=77;
    outtextxy(75,50,message);
    rectangle(75,75,300,100);
    line(cursor,96,cursor+csize,96);
    repeat
      ch:=getkey;
      case ch of
        #8,leftarrow : begin
                         if cursor >= 80+csize then
                         begin
                           setcolor(0);
                           line(cursor,96,cursor+csize,96);
                           cursor:=cursor-csize;
                           outtextxy(cursor,y,t[length(t)]);
                           setcolor(1);
                           line(cursor,96,cursor+csize,96);
                           delete(t,length(t),1);
                         end;
                       end;

        rightarrow, uparrow, downarrow, #9, #186..#196:;


        return : begin
                   getinput:=t;
                   putimage(65,40,replace^,normalput);
                 end;

        Esc : begin
                getinput:='';
                putimage(65,40,replace^,normalput);
              end;


        else  begin
                if cursor <= 80+csize*16 then
                begin
                  t:=t+ch;
                  setcolor(0);
                  line(cursor,96,cursor+csize,96);
                  setcolor(1);
                  outtextxy(cursor,y,ch);
                  cursor:=cursor+csize;
                  line(cursor,96,cursor+csize,96);
                end;
              end;

      end;
```

```
    until (ch=return) or (ch=esc);
  end
  else
  begin
    getmem(replace,imagesize(58,23,294,86));
    getimage(58,23,294,86,replace^);
    putimage(58,23,blank^,normalput);
    settextstyle(smallfont,0,5);
    csize := textwidth('A');
    t:='';
    cursor:=71; y:=44;
    outtextxy(67,29,message);
    rectangle(67,43,267,58);
    line(cursor,55,cursor+csize,55);
    repeat
      ch:=getkey;
      case ch of
        #8,leftarrow : begin
                         if cursor >= 71+csize then
                         begin
                           setcolor(0);
                           line(cursor,55,cursor+csize,55);
                           cursor:=cursor-csize;
                           outtextxy(cursor,y,t[length(t)]);
                           setcolor(1);
                           line(cursor,55,cursor+csize,55);
                           delete(t,length(t),1);
                         end;
                       end;

        rightarrow, uparrow, downarrow, #9, #186..#196:;


        return : begin
                   getinput:=t;
                   putimage(58,23,replace^,normalput);
                 end;

        Esc : begin
                getinput:='';
                putimage(58,23,replace^,normalput);
              end;


        else  begin
                if cursor <= 71+csize*22 then
                begin
                  t:=t+ch;
                  setcolor(0);
                  line(cursor,55,cursor+csize,55);
                  setcolor(1);
                  outtextxy(cursor,y,ch);
                  cursor:=cursor+csize;
                  line(cursor,55,cursor+csize,55);
                end;
              end;

      end;
    until (ch=return) or (ch=esc);
  end;
  settextstyle(0,0,1);
  dispose(replace);
end;
```

```
procedure select(var choice:chtype);

(*  Procedure to select a choice in graphics mode from a menu with
    a scroll bar.                                                    *)

var x,y:integer;
    ch:char;

begin
  x:=290; y:=141;
  cleardevice;
  settextstyle(gothicfont,horizdir,9) ;
  outtextxy(175,5,'Spectrum');
  rectangle(275,125,425,275);
  settextstyle(smallfont,0,5);
  outtextxy(295,140,'A/D Converter');
  outtextxy(295,155,'Load Spectrum');
  outtextxy(295,170,'Save Spectrum');
  outtextxy(295,185,'Load Ascii');
  outtextxy(295,200,'Save Ascii');
  outtextxy(295,215,'Review Graph');
  outtextxy(295,230,'OS shell');
  outtextxy(295,245,'Quit');
  putimage(x,y,sbar^,xorput);
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  putimage(x,y,sbar^,xorput);
                  if y=141 then y:=246
                  else y:=y-15;
                  putimage(x,y,sbar^,xorput);
                end;

      downarrow : begin
                    putimage(x,y,sbar^,xorput);
                    if y=246 then y:=141
                    else y:=y+15;
                    putimage(x,y,sbar^,xorput);
                  end;

      return : case y of
                 141 : choice:=ADconvert;

                 156 : choice:=loadspectrum;

                 171 : choice:=savespectrum;

                 186 : choice:=loadasc;

                 201 : choice:=saveasc;

                 216 : choice:=review;

                 231 : choice:=OSshell;

                 246 : choice:=quit;
               end;

    end;
  until ch=return;
```

```
end;  { select }

procedure pause;       { Pause until keypressed. }
var ch : char;
begin
  repeat until keypressed;
  ch := readkey;
end;

procedure savedata(plot:artype; legend:legrec);

(*  Saves 'plot' and 'legend' to a file of filerec. (non ascii) .   *)

var i:integer;
    fname : string;
    saverec : filerec;
    f:searchrec;
    ch: char;

begin
  ch:='y';
  fname:=getinput('File to save:');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      outtext('Do you want to overwrite '+fname+'? (Y/N) ');
      ch:=null;
      while not(ch in ['y','Y','n','N']) do
        ch:=getkey;
    end;
    if ch in ['y','Y'] then
    begin
      saverec.plot:=plot;
      saverec.legend:=legend;
      assign(fileout,fname);
      rewrite(fileout);
      write(fileout,saverec);
      close(fileout);
    end;
  end;
end;


procedure readdata(var plot:artype; var legend:legrec; var err:boolean; message:
string);

(*  Retrieves a file of filerec. (plot, legend. (non ascii)).    *)

var i:integer;
    fname : string;
    restorerec : filerec;
    f : searchrec;

begin
  fname:=getinput(message);
  if fname <> '' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
```

```
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      assign(filein,fname);
      reset(filein);
      read(filein,restorerec);
      plot:=restorerec.plot;
      legend:=restorerec.legend;
      close(filein);
      err:=false;
    end
    else
    begin
      if doserror<>0 then
      begin
        outtextxy(20,20,'Sorry, but '+fname+' was not found.');
        pause;
        setcolor(0);
        outtextxy(20,20,'Sorry, but '+fname+' was not found.');
      end
      else
      begin
        outtextxy(20,20,fname+' is not a Spectrum file.');
        pause;
        setcolor(0);
        outtextxy(20,20,fname+' is not a Spectrum file.');
      end;
      setcolor(1);
      err:=true;
    end;
  end
  else err:=true;
end;

procedure loadascii(var plot:artype; var err:boolean);

{  Reads an ascii file containing integers to store in 'plot'.  }

var i:integer;
    fname:string;
    f : searchrec;

begin
  fname:=getinput('File to read:');
  if fname <> '' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      assign(inascii,fname);
      reset(inascii);
      for i:=0 to limit do
      if not eof(inascii) then
        read(inascii,plot[i])
      else
        plot[i]:=0;
      close(inascii);
      err:=false;
    end
    else
    begin
      outtext('Sorry, but '+fname+' was not found.');
```

```
      err:=true;
      pause;
    end;
  end
  else err:=true;
end;


procedure saveascii(plot:artype);

{  Stores array "plot" to an ascii file. }

var i:integer;
    fname:string;
    ch:char;
    f : searchrec;

begin
  ch:='y';
  fname:=getinput('File to save:');
  if fname <> '' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      outtext('Do you want to overwrite '+fname+'? (Y/N) ');
      ch:=null;
      while not(ch in ['y','Y','n','N']) do
        ch:=getkey;
    end;
    if ch in ['y','Y'] then
    begin
      assign(outascii,fname);
      rewrite(outascii);
      for i:=0 to limit do
      begin
        write(outascii,plot[i]);
        write(outascii,' ');
      end;
      close(outascii);
    end;
  end;
end;

procedure scrsave;

{  Saves the screen to a file. Can only be viewed, not processed.  }

type scrblock = array[0..31326] of byte;
var fname : string;
    screen: ^scrblock;
    scrfile : file of scrblock;

begin
  fname:=getinput('Save Screen:');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    getmem(screen,31327);
    getimage(0,0,719,347, screen^);
    assign(scrfile,fname);
    rewrite(scrfile);
```

```
    write(scrfile,screen^);
    close(scrfile);
    dispose(screen);
  end;
end;

procedure scrrestore;

{  Lets user view a file which was saved with "scrsave".  }

type scrblock = array[0..31326] of byte;
var fname : string;
    screen,oldscreen: ^scrblock;
    f : searchrec;
    scrfile : file of scrblock;

begin
  fname:=getinput('Restore what screen?');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile,f);
    if (doserror=0) and (f.attr and directory = 0) and (f.size=31327) then
    begin
      assign(scrfile,fname);
      reset(scrfile);
      getmem(oldscreen,31327);
      getimage(0,0,719,347, oldscreen^);
      getmem(screen,31327);
      read(scrfile,screen^);
      close(scrfile);
      putimage(0,0,screen^,normalput);
      dispose(screen);
      pause;
      putimage(0,0,oldscreen^,normalput);
      dispose(oldscreen);
    end
    else
    begin
      if doserror<>0 then
      begin
        outtextxy(20,20,'Sorry, but '+fname+' was not found.');
        pause;
        setcolor(0);
        outtextxy(20,20,'Sorry, but '+fname+' was not found.');
      end
      else
      begin
        outtextxy(20,20,fname+' is not a screen file!');
        pause;
        setcolor(0);
        outtextxy(20,20,fname+' is not a screen file!');
      end;
      setcolor(1);
    end;
  end;
end;

procedure scrsavecga;
type scrblock = array[0..16287] of byte;
var fname : string;
    screen: ^scrblock;
    scrfile : file of scrblock;
```

```
begin
  fname:=getinput('Save Screen:');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    getmem(screen,16288);
    getimage(0,0,639,199, screen^);
    assign(scrfile,fname);
    rewrite(scrfile);
    write(scrfile,screen^);
    close(scrfile);
    dispose(screen);
  end;
end;

procedure scrrestorecga;
type scrblock = array[0..16287] of byte;
var fname : string;
    screen,oldscreen: ^scrblock;
    f : searchrec;
    scrfile : file of scrblock;

begin
  fname:=getinput('Restore what screen?');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile,f);
    if (doserror=0) and (f.attr and directory = 0) and (f.size=16288) then
    begin
      assign(scrfile,fname);
      reset(scrfile);
      getmem(oldscreen,16288);
      getimage(0,0,639,199, oldscreen^);
      getmem(screen,16288);
      read(scrfile,screen^);
      close(scrfile);
      putimage(0,0,screen^,normalput);
      dispose(screen);
      pause;
      putimage(0,0,oldscreen^,normalput);
      dispose(oldscreen);
    end
    else
    begin
      if doserror<>0 then
      begin
        outtextxy(20,20,'Sorry, but '+fname+' was not found.');
        pause;
        setcolor(0);
        outtextxy(20,20,'Sorry, but '+fname+' was not found.');
      end
      else
      begin
        outtextxy(20,20,fname+' is not a screen file!');
        pause;
        setcolor(0);
        outtextxy(20,20,fname+' is not a screen file!');
      end;
      setcolor(1);
    end;
  end;
```

```
end;

function min(x,y:longint):longint;

{ Returns min of two integers }

begin
  if x<y then
    min:=x
  else
    min:=y;
end;

function max(x,y:longint):longint;

{ Returns max of two integers }

begin
  if x>y then
    max:=x
  else
    max:=y;
end;

function fstr(num:integer):string;

{  To make TP 'str' procedure a function. }

var s:string;
begin
  str(num,s);
  fstr:=s;
end;


procedure getuserinp(var t:word; var legend:legrec);

{  Get users input for duration of the sample and the title of the
   graph. Also stores the current date.                              }

var code:integer;
begin
  legend.title:= getinput('Enter graph title:');
  val(getinput('Duration of sample:'), t, code);

  if t > 0 then
    t := round(1000/(limit/t))
  else t:=0;
  with legend.date do
    getdate(year,month,date,dayofweek);
end;

procedure getdata(var plot:artype; time:word);

{  Gets data from A/D converter.  }

const a=$210;
      c=$212;
      ct=$213;
var i, portnum, code, value:integer;
begin
  val(getinput('Enter port num: '), portnum, code);
```

```
  while (portnum<=0) or (portnum>8) do
    val(getinput('Must be in 1..8:'), portnum, code);
  dec(portnum);
  outtext('Getting data, please wait...');
  port[ct]:=$80;
  for i:=0 to limit do
  begin
    value := (4095-round(i * (4095/limit))) shl 4;
    port[c]:=$A0;
    port[a]:=hi(value);
    port[c]:=$80;
    port[a]:=lo(value);
    port[c]:=$90;
    port[c]:=$10;
    port[board+portnum]:=0;
    delay(time);
    plot[i]:=port[board];
    if graph_driver = hercmono then
    begin
      if i mod 10=0 then
        outtextxy(i div 10,10,chr(22));
    end
    else
    if i mod 12=0 then
      outtextxy(i div 11,10,chr(22));
  end;
  cleardevice;
end;

procedure getspot(var plot: artype; var spot:integer);

{  Adjusts spot as to put graph on zero.  }

var i:integer;
    oldplot:integer;

begin
  oldplot:=0;
  for i:=limit-600 to limit do
  if plot[i]>oldplot then
  begin
    if graph_driver = hercmono then
      spot:=i-7000
    else
      spot:=i-7190;
    oldplot:=plot[i];
  end;
end;     { getspot }


function rightjustify(astring:string):integer;

{  Used to position title.  }

begin
  rightjustify:=xmax-textwidth(astring)-30;
end;

procedure axis(legend:legrec);

{   Draws x and y numbered axis with legend printed.  }

var i : integer;
```

```
    s : string[1];

begin
  settextstyle(0,0,1);
  line(0, 0, 0, 325);                                    { vertical axis }
  line(0, 315, 719, 315);                                { horizontal axis }
  for i := 0 to 50 do
  begin
    if i mod 5 <> 0 then
      line(i*xaxis,318, i*xaxis,315)
    else
    begin
      line(i*xaxis, 325 , i*xaxis, 315);
      str(9 - i div 5,s);
      outtextxy((i-1)*xaxis+round(xaxis*4.785714286)+7,320,s);
    end;
  end;

  for i:= 63 downto 0 do
  begin
    if i mod 5 <>3 then
      line(3,i*5,0,i*5)
    else
      line(5,i*5,0,i*5);
  end;

  outtextxy(315,335,'delta (ppm)');
  with legend.date do
    outtextxy(5,340,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)+')');
  settextstyle(triplexfont,0,3);
  outtextxy(rightjustify(legend.title),20,legend.title);

end; { axis }


procedure axiscga(legend:legrec);
var i : integer;
    s : string[1];

begin
  settextstyle(0,0,1);
  line(0, 0, 0, 187);                                    { vertical axis }
  line(0, 180, 640, 180);                                { horizontal axis }
  for i := 0 to 50 do
  begin
    if i mod 5 <> 0 then
      line(i*xaxis,182, i*xaxis,180)
    else
    begin
      line(i*xaxis, 186, i*xaxis, 180);
      str(9 - i div 5,s);
      outtextxy((i-1)*xaxis+round(xaxis*4.785714286)+6,182,s);
    end;
  end;

  for i:= 36 downto 0 do
  begin
    if i mod 5 <> 1 then
      line(3,i*5,0,i*5)
    else
      line(5,i*5,0,i*5);
  end;
```

```
    begin
      x:=getx;
      y:=gety;
      setcolor(0);
      outtextxy(10,10,fstr(op));
      setcolor(1);
      outtextxy(10,10,fstr(p));
      moveto(x,y);
    end;
    if count=0 then moveto(round((value)*(719/4095)),adjust(p))
    else lineto(round((value)*(719/4095)),adjust(p));
    inc(count);
  until ch=esc;
  cleardevice;
end;


procedure addtoarray(add:integer; var plot:artype);

{ Used to move graph up and down. }

var i:integer;

begin
  for i:=0 to limit do
    plot[i]:=plot[i]+add;
end;

procedure average(var plot:artype);
var i:integer;

begin
  for i:=0 to limit-2 do
    plot[i]:=(plot[i]+plot[i+1]+plot[i+2]) div 3;
  plot[limit-1]:=plot[limit-2];
  plot[limit]:=plot[limit-2];
end;

procedure draw(plot:artype;spot:integer);

{  Plots graph then calls integration routine. }

var x : integer;

begin
  moveto(0,adjust(plot[0]));
  for x := 1 to limit-spot do
    if x mod xratio = 0 then
      if x+spot >= 0 then
      begin
        if x+spot <= limit then
          lineto(x div xratio, adjust(plot[x+spot]))
      end
      else
        moveto(x div xratio, adjust(plot[0]));

  if graph_driver = hercmono then
    integrate(plot,spot)        { integrate Nmr graph }
  else
    integratecga(plot,spot);

end; { draw }
```

```
procedure sorti(xput:st; arlimit:integer; var sindex:st);

{ Used to return an array of sorted indexes according to the passed array.
  It's used to make sure the answers from couple and getratio don't overlap.}

type hold= record
             xp:integer;
             postn:integer;
           end;

var rec: array[0..20] of hold;
    temprec:hold;
    i,j:integer;

begin
  for i:=0 to arlimit do
  begin
    rec[i].xp:=xput[i];
    rec[i].postn:=i;
  end;

  for i:= arlimit downto 1 do
    for j:= 0 to i-1 do
    if rec[j].xp > rec[j+1].xp then
    begin
      temprec:=rec[j];
      rec[j]:=rec[j+1];
      rec[j+1]:=temprec;
    end;

  for i:=0 to arlimit do
    sindex[i]:=rec[i].postn;

end;


procedure getratio(plot:artype; xleft,xright,spot:integer);

{  Gets ratio of specified regions of the graph.  }

var x : array[1..20] of longint;
    xput, sindex : st;
    r : array[0..20] of real;
    ch : char;
    xcoor,i,index: integer;
    mark1, mark2 : integer;
    divisor : longint;
    display: string;
    xr:real;

begin
  xcoor:=140;
  index:=0;
  divisor:=maxlongint;
  mark1:=-1;
  for i:=0 to 20 do x[i]:=0;
  putimage(xcoor-3,317,arrow^,xorput);
  repeat
    ch:=getkey;
    case ch of
      leftarrow,
      ctrlleft  : if xcoor>3 then
```

```
                begin
                  putimage(xcoor-3,317,arrow^,xorput);
                  if ch=leftarrow then
                    xcoor:=xcoor-1
                  else
                  if xcoor>17 then
                    xcoor:=xcoor-14;
                  putimage(xcoor-3,317,arrow^,xorput);
                end;

    rightarrow,
    ctrlright   : if xcoor<700 then
                  begin
                    putimage(xcoor-3,317,arrow^,xorput);
                    if ch=rightarrow then
                      xcoor:=xcoor+1
                    else
                    if xcoor<700-14 then
                      xcoor:=xcoor+14;
                    putimage(xcoor-3,317,arrow^,xorput);
                  end;


    spacebar : begin
                  putimage(xcoor-3,317,arrow^,xorput);
                  line(xcoor,310,xcoor,320);
                  putimage(xcoor-3,317,arrow^,xorput);
                  if mark1=-1 then
                  begin
                    mark1:=xcoor;
                    sound(500);
                    delay(30);
                    nosound;
                  end
                  else
                  begin
                    mark2:=xcoor;
                    xr:=xratio*(xright-xleft)/700;
                    for i:= round(min(mark1,mark2)*xr) to round(max(mark1,mark2
)*xr) do
                      x[index]:=x[index]+plot[i+xleft*xratio+spot];

                    xput[index]:=(mark1+mark2) div 2;
                    index:=index+1;
                    mark1:=-1;
                    sound(300);
                    delay(30);
                    nosound;
                  end;
                end;

    return : begin
                for i:=0 to index-1 do
                  divisor:=min(x[i],divisor);
                if divisor=0 then divisor:=1;
                for i:=0 to index-1 do
                  r[i]:= x[i] / divisor;
                cleardevice;
                settextstyle(smallfont,0,4);
                sorti(xput,index-1,sindex);
                for i:=1 to index-1 do
                  if xput[sindex[i]] - xput[sindex[i-1]] <= 30 then
                    xput[sindex[i]]:=xput[sindex[i]]+30-(xput[sindex[i]]-xput[s
```

```
index[i-1]]);
                  for i:=0 to index-1 do
                  begin
                    str(r[i]:6:2,display);
                    outtextxy(xput[i]-textwidth(display) div 2, 325, display);
                  end;
                end;

      esc : cleardevice;
    end;
  until (ch=esc) or (ch=return);
end;


procedure couple(plot:artype; r:real);     { Coupling Constants. }
var ans : array[0..20] of real;
    xput,sindex : st;
    ch:char;
    xcoor,i,j,index: integer;
    mark1, mark2: integer;
    display: string;

begin
  xcoor:=140;
  index:=0;
  mark1:=-1;
  putimage(xcoor-1,0,aline^,xorput);
  repeat
    ch:=getkey;
    case ch of
      leftarrow,
      ctrlleft  : if xcoor>3 then
                  begin
                    putimage(xcoor-1,0,aline^,xorput);
                    if ch=leftarrow then
                      xcoor:=xcoor-1
                    else
                    if xcoor>17 then
                      xcoor:=xcoor-14;
                    putimage(xcoor-1,0,aline^,xorput);
                  end;

      rightarrow,
      ctrlright  : if xcoor<700 then
                   begin
                     putimage(xcoor-1,0,aline^,xorput);
                     if ch=rightarrow then
                       xcoor:=xcoor+1
                     else
                     if xcoor<700-14 then
                       xcoor:=xcoor+14;
                     putimage(xcoor-1,0,aline^,xorput);
                   end;


      spacebar : begin
                   putimage(xcoor-1,0,aline^,xorput);
                   line(xcoor,310,xcoor,320);
                   putimage(xcoor-1,0,aline^,xorput);
                   if mark1=-1 then
                   begin
                     mark1:=xcoor;
                     sound(500);
```

```
                    delay(30);
                    nosound;
                  end
                  else
                  begin
                    mark2:=xcoor;
                    ans[index]:=((max(mark1,mark2)/r)*12)-((min(mark1,mark2)/r)
*12);
                    xput[index]:=(mark1+mark2) div 2;
                    index:=index+1;
                    sound(300);
                    delay(30);
                    nosound;
                    mark1:=-1;
                  end;
                end;

      return : begin

                 putimage(xcoor-1,0,aline^,xorput);

                 sorti(xput,index-1,sindex);

                 for i:=1 to index-1 do
                   if xput[sindex[i]] - xput[sindex[i-1]] <= 40 then
                     xput[sindex[i]]:=xput[sindex[i]]+40-(xput[sindex[i]]-xput[s
index[i-1]]);

                 settextstyle(smallfont,0,4);
                 for i:=0 to index-1 do
                 begin
                   str(ans[i]:6:2, display);
                   outtextxy(xput[i]-textwidth(display)+textwidth('Hz'),325,disp
lay+'Hz');
                 end;

               end;

      esc : cleardevice;

    end;
  until (ch=esc) or (ch=return);
end;


procedure zoom(plot:artype; spot:integer; legend:legrec);

{  Zoom in on specified area of graph.  }

label redo;
var ch:char;
    xleft,xright,x,y,i,xmod,sp:integer;
    shift:byte absolute $40:$17;      { Location of shift status }
    r : real;
    s:string[1];

begin
  xleft:=294;
  xright:=322;
  y:=0;
  putimage(xleft,y,aline^,xorput);
  putimage(xright,y,aline^,xorput);
  setlinestyle(0,0,2);
```

```
line(xleft+2,0,xright,0);

repeat
  ch:=getkey;
  case ch of
    '<' : if shift and 2 = 2 then
          begin
            if xleft>14 then
            begin
              putimage(xleft,y,aline^,xorput);
              xleft:=xleft-14;
              line(xleft+2,0,xright,0);
              putimage(xleft,y,aline^,xorput);
            end;
          end
          else
          if shift and 1 = 1 then
            if xright>xleft+14 then
            begin
              setcolor(0);
              line(xleft+2,0,xright,0);
              putimage(xright,y,aline^,xorput);
              xright:=xright-14;
              setcolor(1);
              line(xleft+2,0,xright,0);
              putimage(xright,y,aline^,xorput);
            end;

    '>' : if shift and 2 = 2 then
          begin
            if xleft<xright-14 then
            begin
              setcolor(0);
              line(xleft+2,0,xright,0);
              putimage(xleft,y,aline^,xorput);
              xleft:=xleft+14;
              setcolor(1);
              line(xleft+2,0,xright,0);
              putimage(xleft,y,aline^,xorput);
            end;
          end
          else
          if shift and 1 = 1 then
            if xright<=700 then
            begin
              putimage(xright,y,aline^,xorput);
              xright:=xright+14;
              line(xleft+2,0,xright,0);
              putimage(xright,y,aline^,xorput);
            end;

    ctrlleft : if xleft>14 then
               begin
                 setcolor(0);
                 line(xleft+2,0,xright,0);
                 putimage(xleft,y,aline^,xorput);
                 putimage(xright,y,aline^,xorput);
                 xleft:=xleft-14;
                 xright:=xright-14;
                 setcolor(1);
                 line(xleft+2,0,xright,0);
                 putimage(xleft,y,aline^,xorput);
                 putimage(xright,y,aline^,xorput);
```

```
                end;

      ctrlright : if xright<=700 then
                  begin
                    setcolor(0);
                    line(xleft+2,0,xright,0);
                    putimage(xleft,y,aline^,xorput);
                    putimage(xright,y,aline^,xorput);
                    xleft:=xleft+14;
                    xright:=xright+14;
                    setcolor(1);
                    line(xleft+2,0,xright,0);
                    putimage(xleft,y,aline^,xorput);
                    putimage(xright,y,aline^,xorput);
                  end;


      return : begin
                 cleardevice;                          {             }
Redo :           settextstyle(0,0,1);                  {  Draw Axis  }
                 line(0, 0, 0, 325);                   {             }
                 line(0, 315, 719, 315);
                 sp:=round(700/((xright-xleft)/14));
                 xmod:=5-round(frac(xleft/70) * 5);
                 if xmod=5 then xmod:=0;
                 for i := 0 to round((xright-xleft)/14) do
                 begin
                   if i mod 5 <> xmod then
                     line(i*sp,307, i*sp,315)
                   else
                   begin
                     line(i*sp, 300, i*sp, 315);
                     str(9 - ((xleft-67+i*14) div 70),s);
                     outtextxy(i*sp,320,s);
                   end;
                 end;

                 for i:= 63 downto 0 do
                 begin
                   if i mod 5 <>3 then
                     line(3,i*5,0,i*5)
                   else
                     line(5,i*5,0,i*5);
                 end;

                 outtextxy(315,335,'delta (ppm)');
                 with legend.date do
                   outtextxy(5,340,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)
+')');
                 settextstyle(triplexfont,0,3);
                 outtextxy(rightjustify(legend.title),20,legend.title);


                 r:=xratio*(xright-xleft)/700;         {  Plot Zoom  }
                 moveto(0,adjust(plot[xleft*xratio+spot]));
                 for x:=1 to 700 do
                 if round(x*r+xleft*xratio)+spot <= limit then
                   lineto(x,adjust(plot[round(x*r+xleft*xratio)+spot]));
                 repeat
                   ch:=getkey;
                   case ch of

                     F2 : scrsave;                     {  for saving and     }
                                                       {  restoring screens. }
```

```
                    F3 : scrrestore;

                    'c','C' : begin
                                couple(plot,sp);
                                goto redo;
                              end;

                    'i','I' : begin
                                getratio(plot,xleft,xright,spot);
                                goto redo;
                              end;
                  end;
                until (ch=esc);
                cleardevice;
                setlinestyle(0,0,normwidth);
                axis(legend);
                draw(plot,spot);
              end;

      esc : begin
              setcolor(0);
              line(xleft+2,0,xright,0);
              putimage(xleft,y,aline^,xorput);
              putimage(xright,y,aline^,xorput);
              setcolor(1);
              setlinestyle(0,0,normwidth);
            end;
    end;

  until (ch=esc);

end;


procedure merge(plot:artype; legend:legrec);
var err : boolean;
    mspot : integer;

begin
  readdata(plot,legend,err,'File to merge:');
  if not err then
  begin
    getspot(plot,mspot);
    draw(plot,mspot);
  end;
end;


procedure cross(var plot:artype; var spot:integer; var legend:legrec);
var plot2:artype;
    i:integer;
    err:boolean;

begin
  readdata(plot2, legend, err, 'Cross file:');
  if not err then
  begin
    for i:=0 to limit do
      plot[i]:=(plot[i]+plot2[i]) div 2;
    getspot(plot,spot);
    legend.title:='Cross';
    cleardevice;
```

```
    if graph_driver=hercmono then
      axis(legend)
    else
      axiscga(legend);
    draw(plot, spot);
  end;
end;


procedure movegr(plot:artype; spot:integer; legend:legrec);

{  Main procedure to process graph. }

var ch:char;

begin
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  addtoarray(1,plot);
                  cleardevice;
                  axis(legend);
                  draw(plot,spot);
                end;

      downarrow : begin
                    addtoarray(-1,plot);
                    cleardevice;
                    axis(legend);
                    draw(plot,spot);
                  end;

      leftarrow : if spot < limit-xmax*xratio then
                  begin
                    spot:=spot+30+xratio;
                    cleardevice;
                    axis(legend);
                    draw(plot,spot);
                  end;

      rightarrow : if spot >= 25 then
                   begin
                     spot:=spot-30+xratio;
                     cleardevice;
                     axis(legend);
                     draw(plot,spot);
                   end;

      homekey : begin
                  spot:=0;
                  cleardevice;
                  axis(legend);
                  draw(plot,spot);
                end;

      endkey : begin
                 spot:=limit-xmax*xratio;
                 cleardevice;
                 axis(legend);
                 draw(plot,spot);
               end;
```

```
      ctrlleft : if spot < limit-xmax*xratio-125 then
                 begin
                   spot:=spot+120+xratio;
                   cleardevice;
                   axis(legend);
                   draw(plot,spot);
                 end;

      ctrlright : if spot >= 125 then
                  begin
                    spot:=spot-120+xratio;
                    cleardevice;
                    axis(legend);
                    draw(plot,spot);
                  end;

      'i','I' : begin
                  getratio(plot,0,700,spot);
                  axis(legend);
                  draw(plot,spot);
                end;

      'c','C' : begin
                  couple(plot,xaxis);
                  axis(legend);
                  draw(plot,spot);
                end;

      'z','Z' : zoom(plot, spot, legend);

      'a','A' : begin
                  average(plot);
                  cleardevice;
                  axis(legend);
                  draw(plot,spot);
                end;

      'm','M' : merge(plot,legend);

      'x','X' : cross(plot,spot,legend);

      'f','F' : begin
                  findpeek;
                  axis(legend);
                  draw(plot,spot);
                end;

      F2 : scrsave;

      F3 : scrrestore;

      altM : ;

    end;
  until ch=#27;
end;


procedure initcga;
var size:integer;
begin
  bar(178,56,285,63);
  size:=imagesize(178,56,285,63);
```

```
  getmem(sbar,size);
  getimage(178,56,285,63,sbar^);
  cleardevice;
  moveto(89,14);
  lineto(102,12);
  lineto(92,8);
  lineto(89,14);
  floodfill(93,12,1);
  setlinestyle(0,0,thickwidth);
  line(94,13,101,21);
  line(96,13,103,21);
  size:=imagesize(89,8,104,22);
  getmem(arrow,size);
  getimage(89,8,104,22,arrow^);
  setlinestyle(0,0,normwidth);
  cleardevice;
  line(31,0,31,180);
  size:=imagesize(30,0,32,180);
  getmem(aline,size);
  getimage(30,0,32,180,aline^);
  cleardevice;
  size:=imagesize(58,23,294,86);
  getmem(blank,size);
  getimage(58,23,294,86,blank^);

end;


procedure selectcga(var choice:chtype);
var x,y:integer;
    ch:char;

begin
  x:=256; y:=93;
  cleardevice;
  settextstyle(gothicfont,horizdir,7);
  outtextxy(188,0,'Spectrum');
  rectangle(243,82,376,169);
  settextstyle(smallfont,0,4);
  outtextxy(261,91,'A/D Converter');
  outtextxy(261,100,'Load Spectrum');
  outtextxy(261,109,'Save Spectrum');
  outtextxy(261,118,'Load Ascii');
  outtextxy(261,127,'Save Ascii');
  outtextxy(261,136,'Review Graph');
  outtextxy(261,145,'OS shell');
  outtextxy(261,154,'Quit');
  putimage(x,y,sbar^,xorput);
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  putimage(x,y,sbar^,xorput);
                  if y=93 then y:=156
                  else y:=y-9;
                  putimage(x,y,sbar^,xorput);
                end;

      downarrow : begin
                    putimage(x,y,sbar^,xorput);
                    if y=156 then y:=93
                    else y:=y+9;
                    putimage(x,y,sbar^,xorput);
```

```
                end;

      return : case y of
                 93 : choice:=ADconvert;

                 102 : choice:=loadspectrum;

                 111 : choice:=savespectrum;

                 120 : choice:=loadasc;

                 129 : choice:=saveasc;

                 138 : choice:=review;

                 147 : choice:=OSshell;

                 156 : choice:=quit;
               end;

    end;
  until ch=return;

end;  { selectcga }


procedure getratiocga(plot:artype; xleft,xright,spot:integer);
var x : array[1..20] of longint;
    xput, sindex : st;
    r : array[0..20] of real;
    ch : char;
    xcoor,i,index: integer;
    mark1, mark2 : integer;
    divisor : longint;
    display: string;
    xr : real;

begin
  xcoor:=125;
  index:=0;
  divisor:=maxlongint;
  mark1:=-1;
  for i:=0 to 20 do x[i]:=0;
  putimage(xcoor-3,182,arrow^,xorput);
  repeat
    ch:=getkey;
    case ch of
      leftarrow,
      ctrlleft  : if xcoor>3 then
                  begin
                    putimage(xcoor-3,182,arrow^,xorput);
                    if ch=leftarrow then
                      xcoor:=xcoor-1
                    else
                    if xcoor>20 then
                      xcoor:=xcoor-12;
                    putimage(xcoor-3,182,arrow^,xorput);
                  end;

      rightarrow,
      ctrlright  : if xcoor<600 then
                    begin
                      putimage(xcoor-3,182,arrow^,xorput);
```

```
                    if ch=rightarrow then
                      xcoor:=xcoor+1
                    else
                    if xcoor<600-12 then
                      xcoor:=xcoor+12;
                    putimage(xcoor-3,182,arrow^,xorput);
                  end;


      spacebar : begin
                   putimage(xcoor-3,182,arrow^,xorput);
                   line(xcoor,179,xcoor,184);
                   putimage(xcoor-3,182,arrow^,xorput);
                   if mark1=-1 then
                   begin
                     mark1:=xcoor;
                     sound(500);
                     delay(30);
                     nosound;
                   end
                   else
                   begin
                     mark2:=xcoor;
                     xr:= xratio*(xright-xleft)/600;
                     for i:= round(min(mark1,mark2)*xr) to round(max(mark1,mark2
)*xr) do
                       x[index]:=x[index]+plot[i+xleft*xratio+spot];

                     xput[index]:=(mark1+mark2) div 2;
                     index:=index+1;
                     mark1:=-1;
                     sound(300);
                     delay(30);
                     nosound;
                   end;
                 end;

      return : begin
                 for i:=0 to index-1 do
                   divisor:=min(x[i],divisor);
                 if divisor=0 then divisor:=1;
                 for i:=0 to index-1 do
                   r[i]:= x[i] / divisor;
                 cleardevice;
                 settextstyle(smallfont,0,4);
                 sorti(xput,index-1,sindex);
                 for i:=1 to index-1 do
                   if xput[sindex[i]] - xput[sindex[i-1]] <= 30 then
                     xput[sindex[i]]:=xput[sindex[i]]+30-(xput[sindex[i]]-xput[s
index[i-1]]);
                 for i:=0 to index-1 do
                 begin
                   str(r[i]:6:2,display);
                   outtextxy(xput[i]-textwidth(display) div 2, 187, display);
                 end;
               end;

      esc : cleardevice;
    end;
  until (ch=esc) or (ch=return);
end;
```

```
procedure couplecga(plot:artype; r:real);    { Coupling Constants. }
var ans : array[0..20] of real;
    xput,sindex : st;
    ch:char;
    xcoor,i,j,index: integer;
    mark1, mark2: integer;
    display: string;

begin
  xcoor:=125;
  index:=0;
  mark1:=-1;
  putimage(xcoor-1,0,aline^,xorput);
  repeat
    ch:=getkey;
    case ch of
      leftarrow,
      ctrlleft  : if xcoor>2 then
                  begin
                    putimage(xcoor-1,0,aline^,xorput);
                    if ch=leftarrow then
                      xcoor:=xcoor-1
                    else
                    if xcoor>15 then
                      xcoor:=xcoor-12;
                    putimage(xcoor-1,0,aline^,xorput);
                  end;

      rightarrow,
      ctrlright  : if xcoor<600 then
                   begin
                     putimage(xcoor-1,0,aline^,xorput);
                     if ch=rightarrow then
                       xcoor:=xcoor+1
                     else
                     if xcoor<700-12 then
                       xcoor:=xcoor+12;
                     putimage(xcoor-1,0,aline^,xorput);
                   end;


      spacebar : begin
                   putimage(xcoor-1,0,aline^,xorput);
                   line(xcoor,179,xcoor,184);
                   putimage(xcoor-1,0,aline^,xorput);
                   if mark1=-1 then
                   begin
                     mark1:=xcoor;
                     sound(500);
                     delay(30);
                     nosound;
                   end
                   else
                   begin
                     mark2:=xcoor;
                     ans[index]:=((max(mark1,mark2)/r)*12)-((min(mark1,mark2)/r)
*12);
                     xput[index]:=(mark1+mark2) div 2;
                     index:=index+1;
                     sound(300);
                     delay(30);
                     nosound;
                     mark1:=-1;
```

```
                    end;
                  end;

      return : begin

                  putimage(xcoor-1,0,aline^,xorput);

                  sorti(xput,index-1,sindex);

                  for i:=1 to index-1 do
                    if xput[sindex[i]] - xput[sindex[i-1]] <= 40 then
                      xput[sindex[i]]:=xput[sindex[i]]+40-(xput[sindex[i]]-xput[s
index[i-1]]);

                  settextstyle(smallfont,0,4);
                  for i:=0 to index-1 do
                  begin
                    str(ans[i]:6:2, display);
                    outtextxy(xput[i]-textwidth(display)+textwidth('Hz'),187,disp
lay+'Hz');
                  end;

                end;

      esc : cleardevice;

    end;
  until (ch=esc) or (ch=return);
end;


procedure zoomcga(plot:artype; spot:integer; legend:legrec);

label redo;
var ch:char;
    xleft,xright,x,y,i,xmod,sp:integer;
    shift : byte absolute 0:$417;
    r : real;
    s:string[1];

begin
  xleft:=264;
  xright:=288;
  y:=0;
  putimage(xleft,y,aline^,xorput);
  putimage(xright,y,aline^,xorput);
  setlinestyle(0,0,2);
  line(xleft+2,0,xright-1,0);

  repeat
    ch:=getkey;
    case ch of
      '<' : if shift and 2 = 2 then
            begin
              if xleft>12 then
              begin
                putimage(xleft,y,aline^,xorput);
                xleft:=xleft-12;
                line(xleft+2,0,xright-1,0);
                putimage(xleft,y,aline^,xorput);
              end;
            end
            else
```

```
        if shift and 1 = 1 then
          if xright>xleft+12 then
          begin
            setcolor(0);
            line(xleft+2,0,xright-1,0);
            putimage(xright,y,aline^,xorput);
            xright:=xright-12;
            setcolor(1);
            line(xleft+2,0,xright-1,0);
            putimage(xright,y,aline^,xorput);
          end;


'>' : if shift and 2 = 2 then
        begin
          if xleft<xright-12 then
          begin
            setcolor(0);
            line(xleft+2,0,xright-1,0);
            putimage(xleft,y,aline^,xorput);
            xleft:=xleft+12;
            setcolor(1);
            line(xleft+2,0,xright-1,0);
            putimage(xleft,y,aline^,xorput);
          end;
          end
          else
          if shift and 1 = 1 then
            if xright<=600 then
            begin
              putimage(xright,y,aline^,xorput);
              xright:=xright+12;
              line(xleft+2,0,xright-1,0);
              putimage(xright,y,aline^,xorput);
            end;


ctrlleft : if xleft>12 then
            begin
              setcolor(0);
              line(xleft+2,0,xright-1,0);
              putimage(xleft,y,aline^,xorput);
              putimage(xright,y,aline^,xorput);
              xleft:=xleft-12;
              xright:=xright-12;
              setcolor(1);
              line(xleft+2,0,xright-1,0);
              putimage(xleft,y,aline^,xorput);
              putimage(xright,y,aline^,xorput);
            end;


ctrlright : if xright<=600 then
             begin
               setcolor(0);
               line(xleft+2,0,xright-1,0);
               putimage(xleft,y,aline^,xorput);
               putimage(xright,y,aline^,xorput);
               xleft:=xleft+12;
               xright:=xright+12;
               setcolor(1);
               line(xleft+2,0,xright-1,0);
```

```
                putimage(xleft,y,aline^,xorput);
                putimage(xright,y,aline^,xorput);
              end;


      return : begin
                 cleardevice;
Redo :           settextstyle(0,0,1);
                 line(0, 0, 0, 187);
                 line(0, 180, 640, 180);
                 sp:=round(600/((xright-xleft)/12));
                 xmod:=5-round(frac(xleft/60) * 5);
                 if xmod=5 then xmod:=0;
                 for i := 0 to round((xright-xleft)/12) do
                 begin
                   if i mod 5 <> xmod then
                     line(i*sp,175, i*sp,180)
                   else
                   begin
                     line(i*sp, 171, i*sp, 180);
                     str(9 - ((xleft-57+i*12) div 60),s);
                     outtextxy(i*sp,182,s);
                   end;
                 end;

                 for i:= 36 downto 0 do
                 begin
                   if i mod 5 <> 1 then
                     line(3,i*5,0,i*5)
                   else
                     line(5,i*5,0,i*5);
                 end;

                 outtextxy(280,192,'delta (ppm)');
                 with legend.date do
                   outtextxy(4,192,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)
+')');
                 settextstyle(triplexfont,0,2);
                 outtextxy(rightjustify(legend.title),12,legend.title);

                 r:=xratio*(xright-xleft)/600;
                 moveto(0,adjust(plot[xleft*xratio+spot]));
                 for x:=1 to 600 do
                 if round(x*r+xleft*xratio)+spot <= limit then
                   lineto(x,adjust(plot[round(x*r+xleft*xratio)+spot]));
                 repeat
                   ch:=getkey;
                   case ch of

                     F2 : scrsavecga;

                     F3 : scrrestorecga;

                     'c','C' : begin
                                 couplecga(plot,sp);
                                 goto redo;
                               end;

                     'i','I' : begin
                                 getratiocga(plot,xleft,xright,spot);
                                 goto redo;
                               end;
```

```
                end;
              until (ch=esc);
              cleardevice;
              setlinestyle(0,0,normwidth);
              axiscga(legend);
              draw(plot,spot);
            end;

      esc : begin
              setcolor(0);
              line(xleft+2,0,xright-1,0);
              putimage(xleft,y,aline^,xorput);
              putimage(xright,y,aline^,xorput);
              setcolor(1);
              setlinestyle(0,0,normwidth);
            end;
    end;

  until (ch=esc);

end;


procedure movegrcga(plot:artype; spot:integer; legend:legrec);
var ch:char;

begin
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  addtoarray(1,plot);
                  cleardevice;
                  axiscga(legend);
                  draw(plot,spot);
                end;

      downarrow : begin
                    addtoarray(-1,plot);
                    cleardevice;
                    axiscga(legend);
                    draw(plot,spot);
                  end;

      leftarrow : if spot < limit-xmax*xratio then
                  begin
                    spot:=spot+30+xratio;
                    cleardevice;
                    axiscga(legend);
                    draw(plot,spot);
                  end;

      rightarrow : if spot >= 25 then
                   begin
                     spot:=spot-30+xratio;
                     cleardevice;
                     axiscga(legend);
                     draw(plot,spot);
                   end;

      homekey : begin
                  spot:=0;
                  cleardevice;
```

```
                axiscga(legend);
                draw(plot,spot);
              end;

    endkey : begin
               spot:=limit-xmax*xratio;
               cleardevice;
               axiscga(legend);
               draw(plot,spot);
             end;

    ctrlleft : if spot < limit-xmax*xratio-125 then
               begin
                 spot:=spot+120+xratio;
                 cleardevice;
                 axiscga(legend);
                 draw(plot,spot);
               end;

    ctrlright : if spot >= 125 then
                begin
                  spot:=spot-120+xratio;
                  cleardevice;
                  axiscga(legend);
                  draw(plot,spot);
                end;

    'i','I' : begin
                getratiocga(plot,0,600,spot);
                axiscga(legend);
                draw(plot,spot);
              end;

    'c','C' : begin
                couplecga(plot,xaxis);
                axiscga(legend);
                draw(plot,spot);
              end;

    'a','A' : begin
                average(plot);
                cleardevice;
                axiscga(legend);
                draw(plot,spot);
              end;

    'z','Z' : zoomcga(plot, spot, legend);

    'm','M' : merge(plot, legend);

    'x','X' : cross(plot,spot,legend);

    F2 : scrsavecga;

    F3 : scrrestorecga;

    altM : ;
   end;
 until ch=#27;
end;

{ ************************************************************************************
```

```
*                              Main Block                                    *
*****************************************************************************}

begin

  initialize;

  if graph_driver = hercmono then
  begin
    initherc;

    repeat
      spot:=0;
      xratio:=10; yratio:=255;
      xaxis:=14;
      select(choice);                                  { select input source }
      case choice of
        ADconvert : begin
                      cleardevice;
                      getuserinp(time,legend);
                      getdata(plot,time);
                      axis(legend);
                      getspot(plot,spot);
                      draw(plot,spot);
                      movegr(plot,spot,legend);
                    end;

        loadspectrum : begin
                         cleardevice;
                         readdata(plot,legend,error,'File to read:');
                         if not error then
                         begin
                           cleardevice;
                           axis(legend);
                           getspot(plot,spot);
                           draw(plot,spot);
                           movegr(plot,spot,legend);
                         end;
                       end;

        savespectrum : savedata(plot,legend);

        loadasc : begin
                    cleardevice;
                    loadascii(plot,error);
                    if not error then
                    begin
                      cleardevice;
                      legend.title:=getinput('Enter graph title:');
                      with legend.date do
                        getdate(year,month,date,dayofweek);
                      axis(legend);
                      getspot(plot,spot);
                      draw(plot,spot);
                      movegr(plot,spot,legend);
                    end;
                  end;

        saveasc : saveascii(plot);

        review : begin
                   cleardevice;
                   axis(legend);
```

```
                getspot(plot,spot);
                draw(plot,spot);
                movegr(plot,spot,legend);
              end;

      OSshell : begin
                  restorecrtmode;
                  writeln('Type EXIT to return to menu...');
                  swapvectors;
                  exec(getenv('comspec'),'/c '+getenv('comspec'));
                  swapvectors;
                  setgraphmode(graph_mode);
                end;

    end;
  until(choice = quit)
end
else
begin

  initcga;

  repeat
    spot:=0;
    xratio:=12; yratio:=255;
    xaxis:=12;
    selectcga(choice);
    case choice of
      ADconvert : begin
                    cleardevice;
                    getuserinp(time,legend);
                    getdata(plot,time);
                    axiscga(legend);
                    getspot(plot,spot);
                    draw(plot,spot);
                    movegrcga(plot,spot,legend);
                  end;

      loadspectrum : begin
                       cleardevice;
                       readdata(plot,legend,error,'File to read:');
                       if not error then
                       begin
                         cleardevice;
                         axiscga(legend);
                         getspot(plot,spot);
                         draw(plot,spot);
                         movegrcga(plot,spot,legend);
                       end;
                     end;

      savespectrum : savedata(plot,legend);

      loadasc : begin
                  cleardevice;
                  loadascii(plot,error);
                  if not error then
                  begin
                    legend.title:=getinput('Enter graph title:');
                    with legend.date do
                      getdate(year,month,date,dayofweek);
                    axiscga(legend);
                    getspot(plot,spot);
```

```
                        draw(plot,spot);
                        movegrcga(plot,spot,legend);
                      end;
                    end;

        saveasc : saveascii(plot);

        review : begin
                   cleardevice;
                   axiscga(legend);
                   getspot(plot,spot);
                   draw(plot,spot);
                   movegrcga(plot,spot,legend);
                 end;

        OSshell : begin
                    restorecrtmode;
                    writeln('Type EXIT to return to menu...');
                    swapvectors;
                    exec(getenv('comspec'),'/c '+getenv('comspec'));
                    swapvectors;
                    setgraphmode(graph_mode);
                  end;

      end;

    until(choice = quit);

  end;

  closegraph;

end. { main }
```

I.R.

I.R. Interfacing

The Perkin-Elmer 710B Infra Red spectrophotometer was interfaced to the BEST AT computer via the Real Time Devices Inc. ADA300 card. The 0 - 1 volt output from the spectrophotometer was connected to the ADA300 analog to digital converter via the same interface that was used for the NMR set up shown in Appendix 1. Synchronisation of the sweep and sampling procedures was achieved by adjusting the sampling duration to correspond to the sweep duration and by starting the two processes at the same time. In other words the frequency sweep was not controlled from the computer. Technical details are given in Appendix 2.

A program, Infrared, was written in Turbo Pascal to gather data from the IR and display it graphically.

The program uses ports 200-208 hex, which are assigned to the A/D converter, to gather data into an array of 7190 integers (procedure getdata). It then plots every 10 of these values because 719 is the maximum width of the Hercules screen.

For zooming we decrement this ratio with a function such as:

r = 10 * (rightmark - leftmark) / 700

The index is incremented by one and multiplied by r, added to leftmark and plotted until the index reaches the limit.

Infrared

This program is designed to graph data produced by the IR or other instruments with the use of the A to D converter.

Graphics Adapters supported: Hercules, CGA.

Run time commands for Infrared:

From the main menu use the arrow keys and press return to select your choice.

A/D Converter is to obtain the data directly from the Nmr. Be sure to specify the correct port number.

Load IR loads the data that was saved with Save IR and plots the graph.

Save IR saves the data, the title and the date to a file.

Load Ascii loads an ascii file containing integers to be plotted.

Save Ascii saves the data to an ascii file.

Review Graph lets you view the graph that is currently in memory.

OS shell lets you exit to DOS to perform some tasks.

Quit exits the program.

Processing the graph.

Zoom: In the graph area, press 'Z' to zoom a specified area of the graph:

left shift + '<' moves left side left.
left shift + '>' moves left side right.
right shift + '<' moves right side left.
right shift + '>' moves right side right.
ctrl + arrow keys moves entire box.
return to select.

From whithin a zoom you can save the screen (F2) and view a screen (F3).

Absorbance: Press 'A' to plot the absorbance.

Compare : Press 'C' to view more than one graph at the same time.

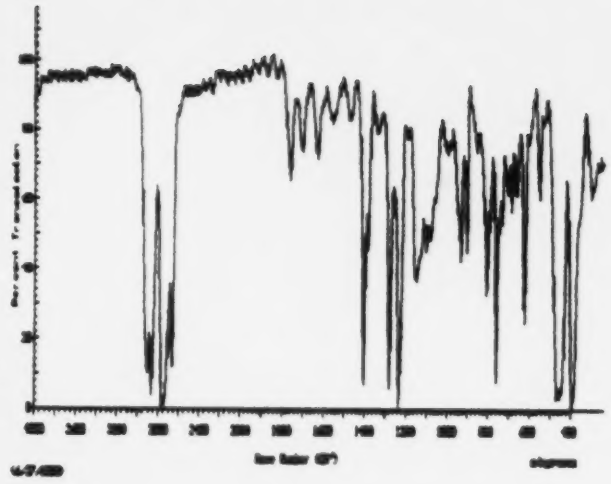
Remove : Preess 'R' to remove a graph that was loaded with compare.

Moving the graph : Use the Up & Down arrow keys to move the graph.

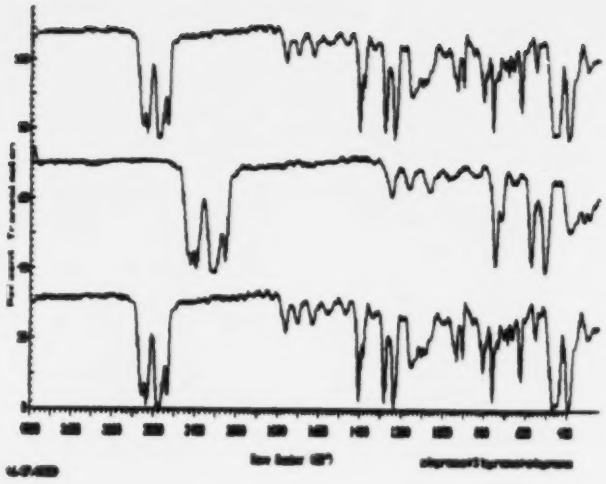
Screen save: F2 to save the screen for quick reviewing.

Screen restore: F3 to restore the screen that was saved with F2.

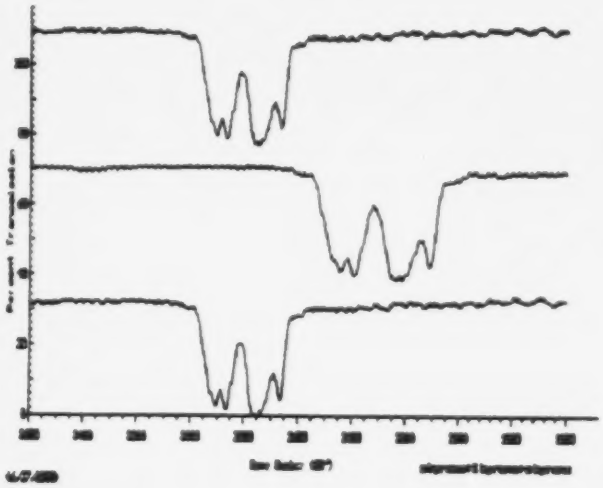
Exit: ESC to exit to main menu.



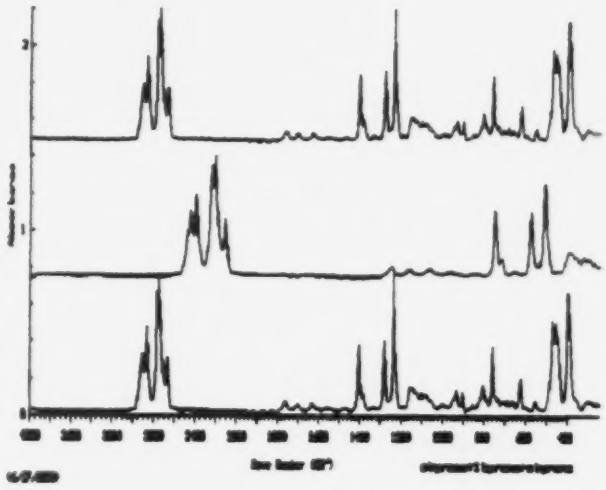
1. Styrene



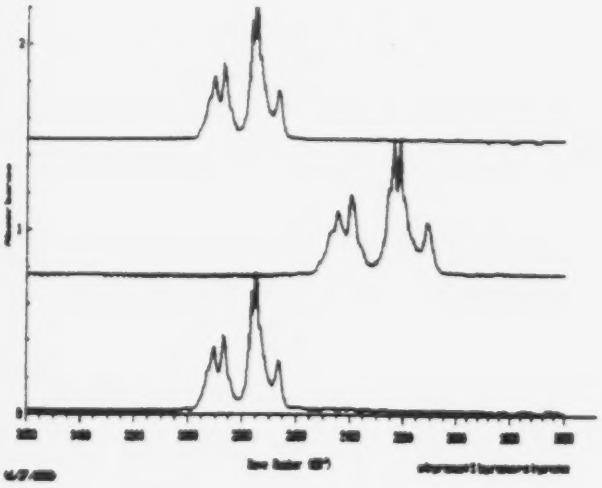
2. Compare



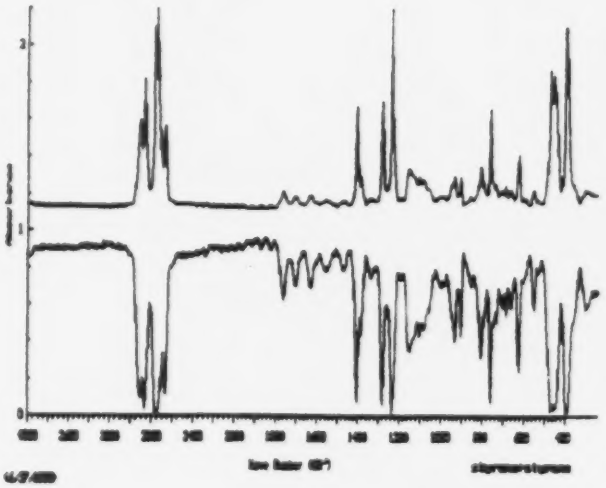
3. Zoom



4. Absorbance



5. Zoom (absorbance)



6. Compare Absorbance + Transmittance

```
program IR;

{*****************************************************************************
```

This program is designed to graph data produced by the IR or other instruments with the use of the A to D converter.

Graphics Adapters supported: Hercules, CGA.

Run time commands for Infrared:

From the main menu use the arrow keys and press return to select your choice.

A/D Converter is to obtain the data directly from the Nmr. Be sure to specify the correct port number.

Load IR loads the data that was saved with Save IR and plots the graph.

Save IR saves the data, the title and the date to a file.

Load Ascii loads an ascii file containing integers to be plotted.

Save Ascii saves the data to an ascii file.

Review Graph lets you view the graph that is currently in memory.

OS shell lets you exit to DOS to perform some tasks.

Quit exits the program.

Processing the graph.

Zoom: In the graph area, press 'Z' to zoom a specified area of the graph:

left shift + '<' moves left side left.
left shift + '>' moves left side right.
right shift + '<' moves right side left.
right shift + '>' moves right side right.
ctrl + arrow keys moves entire box.
return to select.

From whithin a zoom you can save the screen (F2) and view a screen (F3).

Absorbance: Press 'A' to plot the absorbance.

Compare : Press 'C' to view more than one graph at the same time.

```
      Remove :   Preess 'R' to remove a graph that was loaded with compare.
 Moving the graph :   Use the Up & Down arrow keys to move the graph.
   Screen save:   F2 to save the screen for quick reviewing.
Screen restore:   F3 to restore the screen that was saved with F2.
          Exit:   ESC to exit to main menu.
```


Colin Lamarre
John Abbott College
June 17, 1989.


```
*****************************************************************************}

{$M 64512,0,129024}     { memory sizes }

uses crt, dos, graph, drivers, fonts;    { TP 5.0 units needed }

const
     uparrow=#200;           { Define keys }
     downarrow=#208;
     leftarrow=#203;
     rightarrow=#205;
     homekey=#199;
     endkey=#207;
     ctrlleft=#243;
     ctrlright=#244;
     altM=#178;
     spacebar=#32;
     esc=#27;
     return=#13;
     F1=#187;
     F2=#188;
     F3=#189;
     board = $200;       { Port address }
     null=#0;
     limit=6890;         { array size and amount of data gathered }

type chtype=(ADconvert,loadIR,saveIR,loadasc,saveasc,savescreen,restorescreen,re
view,OSshell,quit);
     artype= array[0..limit] of integer;   { array to store digital values }
     plots = array[1..3] of artype;
     st=array[0..20] of integer;
     daterec=record
               year,month,date,dayofweek:word;
             end;
     legrec=record                  { record to hold legend }
              title: string;
              date : daterec;
            end;
     filerec=record              { used for file I/O }
               plot:artype;
               legend:legrec;
             end;
```

```
var  graph_driver  :  integer;      { to initialize graph driver & mode }
     graph_mode    :  integer;
     choice        :  chtype;       { to store user's choice }
     plot : plots;                  { digital values to be plotted }
     time : word;                   { duration of sample }
     Xmax, Ymax : integer;          { max vals for screen type }
     legend : legrec;
     filein, fileout : file of filerec;    { file I/O }
     inascii,outascii : text;
     error : boolean;
     sbar, arrow, aline, blank : pointer;  { to store images }
     xratio, yratio : integer;             { to move, expand graph }
     ybord:integer;
     numberofarrays : 0..3;            { Global }
     absorb:boolean;
     oldtitle:array[2..3] of string;


procedure initialize;

(*   Links all graph drivers and fonts into EXE file to
     produce a stand alone program.                          *)

begin
  if registerbgidriver(@cgadriverproc) < 0 then writeln('error');
  if registerbgidriver(@hercdriverproc) < 0 then writeln('error');
  if registerbgifont(@triplexfontproc) < 0 then writeln('error');
  if registerbgifont(@smallfontproc) < 0 then writeln('error');
  if registerbgifont(@gothicfontproc) < 0 then writeln('error');
  detectgraph(graph_driver,graph_mode);
  initgraph(graph_driver, graph_mode, '');
  xmax:=getmaxx; ymax:=getmaxy;
  if graph_driver=hercmono then ybord:=30 else ybord:=27;
  checkbreak:=false;
end;


procedure initherc;

(* Gets all images used by Hercules adapter.     *)

var size:integer;
begin
  bar(200,100,320,110);
  size:=imagesize(200,100,320,110);
  getmem(sbar,size);
  getimage(200,100,320,110,sbar^);
  cleardevice;
  moveto(100,25);
  lineto(115,21);
  lineto(103,14);
  lineto(100,25);
  floodfill(105,20,1);
  setlinestyle(0,0,thickwidth);
  line(106,23,114,37);
  line(108,22,116,36);
  size:=imagesize(100,14,116,37);
  getmem(arrow,size);
  getimage(100,14,117,37,arrow^);
  setlinestyle(0,0,normwidth);
  cleardevice;
  line(35,0,35,290);
```

```
  size:=imagesize(34,0,36,290);
  getmem(aline,size);
  getimage(34,0,36,290,aline^);
  cleardevice;
  size:=imagesize(65,40,330,150);
  getmem(blank,size);
  getimage(65,40,330,150,blank^);
end;


function GetKey:char;

(*  Returns key value. *)

var
  C : Char;
begin
  C := ReadKey;
  repeat
    if C = NULL then
    begin
      C := ReadKey;
      if Ord(C) > 127 then
        C := NULL
      else
        GetKey := Chr(Ord(C) + 128);
    end
    else
      GetKey := C;
  until C <> NULL;
end; { GetKey }


function getinput(message:string):string;

(* Lets user type input in graphics mode. It saves the screen section
   then replaces it when done.                                        *)

var ch : char;
    cursor,csize,y : integer;
    t : string;
    replace:pointer;

begin
  if graph_driver = hercmono then
  begin
    getmem(replace,imagesize(65,40,330,150));
    getimage(65,40,330,150,replace^);
    putimage(65,40,blank^,normalput);
    settextstyle(smallfont,0,7);
    csize := textwidth('A');
    t:='';
    cursor:=80; y:=77;
    outtextxy(75,50,message);
    rectangle(75,75,300,100);
    line(cursor,96,cursor+csize,96);
    repeat
      ch:=getkey;
      case ch of
        #8,leftarrow : begin
                         if cursor >= 80+csize then
                         begin
                           setcolor(0);
```

```
                        line(cursor,96,cursor+csize,96);
                        cursor:=cursor-csize;
                        outtextxy(cursor,y,t[length(t)]);
                        setcolor(1);
                        line(cursor,96,cursor+csize,96);
                        delete(t,length(t),1);
                      end;
                    end;

      rightarrow, uparrow, downarrow, #9, #186..#196:;


      return : begin
                 getinput:=t;
                 putimage(65,40,replace^,normalput);
               end;

      Esc : begin
              getinput:='';
              putimage(65,40,replace^,normalput);
            end;


      else  begin
              if cursor <= 80+csize*16 then
              begin
                t:=t+ch;
                setcolor(0);
                line(cursor,96,cursor+csize,96);
                setcolor(1);
                outtextxy(cursor,y,ch);
                cursor:=cursor+csize;
                line(cursor,96,cursor+csize,96);
              end;
            end;

    end;
  until (ch=return) or (ch=esc);
end
else
begin
  getmem(replace,imagesize(58,23,294,86));
  getimage(58,23,294,86,replace^);
  putimage(58,23,blank^,normalput);
  settextstyle(smallfont,0,5);
  csize := textwidth('A');
  t:='';
  cursor:=71; y:=44;
  outtextxy(67,29,message);
  rectangle(67,43,267,58);
  line(cursor,55,cursor+csize,55);
  repeat
    ch:=getkey;
    case ch of
      #8,leftarrow : begin
                       if cursor >= 71+csize then
                       begin
                         setcolor(0);
                         line(cursor,55,cursor+csize,55);
                         cursor:=cursor-csize;
                         outtextxy(cursor,y,t[length(t)]);
                         setcolor(1);
                         line(cursor,55,cursor+csize,55);
```

```
                        delete(t,length(t),1);
                      end;
                    end;

      rightarrow, uparrow, downarrow, #9, #186..#196:;


      return : begin
                 getinput:=t;
                 putimage(58,23,replace^,normalput);
               end;

      Esc : begin
              getinput:='';
              putimage(58,23,replace^,normalput);
            end;


      else  begin
              if cursor <= 71+csize*22 then
              begin
                t:=t+ch;
                setcolor(0);
                line(cursor,55,cursor+csize,55);
                setcolor(1);
                outtextxy(cursor,y,ch);
                cursor:=cursor+csize;
                line(cursor,55,cursor+csize,55);
              end;
            end;

    end;
  until (ch=return) or (ch=esc);
 end;
 settextstyle(0,0,1);
 dispose(replace);
end;


procedure select(var choice:chtype);

(*  Procedure to select a choice in graphics mode from a menu with
    a scroll bar.                                                    *)

var x,y:integer;
    ch:char;

begin
  x:=290; y:=141;
  cleardevice;
  settextstyle(gothicfont,horizdir,9) ;
  outtextxy(185,5,'Infrared');
  rectangle(275,125,425,275);
  settextstyle(smallfont,0,5);
  outtextxy(295,140,'A/D Converter');
  outtextxy(295,155,'Load IR');
  outtextxy(295,170,'Save IR');
  outtextxy(295,185,'Load Ascii');
  outtextxy(295,200,'Save Ascii');
  outtextxy(295,215,'Review Graph');
  outtextxy(295,230,'OS shell');
  outtextxy(295,245,'Quit');
  putimage(x,y,sbar^,xorput);
```

```
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  putimage(x,y,sbar^,xorput);
                  if y=141 then y:=246
                  else y:=y-15;
                  putimage(x,y,sbar^,xorput);
                end;

      downarrow : begin
                    putimage(x,y,sbar^,xorput);
                    if y=246 then y:=141
                    else y:=y+15;
                    putimage(x,y,sbar^,xorput);
                  end;

      return : case y of
                 141 : choice:=ADconvert;

                 156 : choice:=loadIR;

                 171 : choice:=saveIR;

                 186 : choice:=loadasc;

                 201 : choice:=saveasc;

                 216 : choice:=review;

                 231 : choice:=OSshell;

                 246 : choice:=quit;
               end;

    end;
  until ch=return;

end;  { select }

procedure pause;       { Pause until keypressed. }
var ch : char;
begin
  repeat until keypressed;
  ch := readkey;
end;

procedure savedata(splot:artype; legend:legrec);

(*  Saves 'plot' and 'legend' to a file of filerec. (non ascii) .   *)

var i:integer;
    fname : string;
    saverec : filerec;
    f:searchrec;
    ch: char;

begin
  ch:='y';
  fname:=getinput('File to save:');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
```

```
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      outtext('Do you want to overwrite '+fname+'? (Y/N) ');
      ch:=null;
      while not(ch in ['y','Y','n','N']) do
        ch:=getkey;
    end;
    if ch in ['y','Y'] then
    begin
      saverec.plot:=splot;
      saverec.legend:=legend;
      assign(fileout,fname);
      rewrite(fileout);
      write(fileout,saverec);
      close(fileout);
    end;
  end;
end;


procedure readdata(var splot:artype; var legend:legrec; var err:boolean; message
:string);

(*  Retrieves a file of filerec. (plot, legend. (non ascii)).    *)

var i:integer;
    fname : string;
    restorerec : filerec;
    f : searchrec;

begin
  fname:=getinput(message);
  if fname <> '' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      assign(filein,fname);
      reset(filein);
      read(filein,restorerec);
      splot:=restorerec.plot;
      legend:=restorerec.legend;
      close(filein);
      err:=false;
    end
    else
    begin
      if doserror<>0 then
      begin
        outtextxy(40,10,'Sorry, but '+fname+' was not found.');
        pause;
        setcolor(0);
        outtextxy(40,10,'Sorry, but '+fname+' was not found.');
      end
      else
      begin
        outtextxy(40,10,fname+' is not an Infrared file.');
        pause;
        setcolor(0);
        outtextxy(40,10,fname+' is not an Infrared file.');
      end;
```

```
      setcolor(1);
      err:=true;
    end;
  end
  else err:=true;
end;


procedure loadascii(var splot:artype; var err:boolean);

{  Reads an ascii file containing integers to store in 'plot'.  }

var i:integer;
    fname:string;
    f : searchrec;

begin
  fname:=getinput('File to read:');
  if fname <> '' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
      assign(inascii,fname);
      reset(inascii);
      for i:=0 to limit do
      if not eof(inascii) then
        read(inascii,splot[i])
      else
        splot[i]:=0;
      close(inascii);
      err:=false;
    end
    else
    begin
      outtext('Sorry, but '+fname+' was not found.');
      err:=true;
      pause;
    end;
  end
  else err:=true;
end;


procedure saveascii(splot:artype);

{  Stores array "plot" to an ascii file. }

var i:integer;
    fname:string;
    ch:char;
    f : searchrec;

begin
  ch:='y';
  fname:=getinput('File to save:');
  if fname <> '' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile, f);
    if (doserror=0) and (f.attr and directory = 0) then
    begin
```

```
      outtext('Do you want to overwrite '+fname+'? (Y/N) ');
      ch:=null;
      while not(ch in ['y','Y','n','N']) do
        ch:=getkey;
    end;
    if ch in ['y','Y'] then
    begin
      assign(outascii,fname);
      rewrite(outascii);
      for i:=0 to limit do
      begin
        write(outascii,splot[i]);
        write(outascii,' ');
      end;
      close(outascii);
    end;
  end;
end;

procedure scrsave;

{  Saves the screen to a file. Can only be viewed, not processed.  }

type scrblock = array[0..31326] of byte;
var fname : string;
    screen: ^scrblock;
    scrfile : file of scrblock;

begin
  fname:=getinput('Save Screen:');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    getmem(screen,31327);
    getimage(0,0,719,347, screen^);
    assign(scrfile,fname);
    rewrite(scrfile);
    write(scrfile,screen^);
    close(scrfile);
    dispose(screen);
  end;
end;

procedure scrrestore;

{  Lets user view a file which was saved with "scrsave".  }

type scrblock = array[0..31326] of byte;
var fname : string;
    screen,oldscreen: ^scrblock;
    f : searchrec;
    scrfile : file of scrblock;

begin
  fname:=getinput('Restore what screen?');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile,f);
    if (doserror=0) and (f.attr and directory = 0) and (f.size=31327) then
    begin
      assign(scrfile,fname);
      reset(scrfile);
```

```
        getmem(oldscreen,31327);
        getimage(0,0,719,347, oldscreen^);
        getmem(screen,31327);
        read(scrfile,screen^);
        close(scrfile);
        putimage(0,0,screen^,normalput);
        dispose(screen);
        pause;
        putimage(0,0,oldscreen^,normalput);
        dispose(oldscreen);
      end
      else
      begin
        if doserror<>0 then
        begin
          outtextxy(40,10,'Sorry, but '+fname+' was not found.');
          pause;
          setcolor(0);
          outtextxy(40,10,'Sorry, but '+fname+' was not found.');
        end
        else
        begin
          outtextxy(40,10,fname+' is not a screen file!');
          pause;
          setcolor(0);
          outtextxy(40,10,fname+' is not a screen file!');
        end;
        setcolor(1);
      end;
    end;
  end;

procedure scrsavecga;
type scrblock = array[0..16287] of byte;
var fname : string;
    screen: ^scrblock;
    scrfile : file of scrblock;

begin
  fname:=getinput('Save Screen:');
  if fname<>'' then
  begin
    fname:=fexpand(fname);
    getmem(screen,16288);
    getimage(0,0,639,199, screen^);
    assign(scrfile,fname);
    rewrite(scrfile);
    write(scrfile,screen^);
    close(scrfile);
    dispose(screen);
  end;
end;

procedure scrrestorecga;
type scrblock = array[0..16287] of byte;
var fname : string;
    screen,oldscreen: ^scrblock;
    f : searchrec;
    scrfile : file of scrblock;

begin
  fname:=getinput('Restore what screen?');
  if fname<>'' then
```

```
  begin
    fname:=fexpand(fname);
    findfirst(fname,anyfile,f);
    if (doserror=0) and (f.attr and directory = 0) and (f.size=16288) then
    begin
      assign(scrfile,fname);
      reset(scrfile);
      getmem(oldscreen,16288);
      getimage(0,0,639,199, oldscreen^);
      getmem(screen,16288);
      read(scrfile,screen^);
      close(scrfile);
      putimage(0,0,screen^,normalput);
      dispose(screen);
      pause;
      putimage(0,0,oldscreen^,normalput);
      dispose(oldscreen);
    end
    else
    begin
      if doserror<>0 then
      begin
        outtextxy(40,10,'Sorry, but '+fname+' was not found.');
        pause;
        setcolor(0);
        outtextxy(40,10,'Sorry, but '+fname+' was not found.');
      end
      else
      begin
        outtextxy(40,10,fname+' is not a screen file!');
        pause;
        setcolor(0);
        outtextxy(40,10,fname+' is not a screen file!');
      end;
      setcolor(1);
    end;
  end;
end;


function min(x,y:longint):longint;

{ Returns min of two integers }

begin
  if x<y then
    min:=x
  else
    min:=y;
end;

function max(x,y:longint):longint;

{ Returns max of two integers }

begin
  if x>y then
    max:=x
  else
    max:=y;
end;

function fstr(num:integer):string;
```

```
{  To make TP 'str' procedure into a function. }

var s:string;
begin
  str(num,s);
  fstr:=s;
end;


procedure getuserinp(var t:word; var legend:legrec);

{  Get users input for duration of the sample and the title of the
   graph. Also stores the current date.                          }

var code:integer;
begin
  legend.title:= getinput('Enter graph title:');
  val(getinput('Duration of sample:'), t, code);

  if t > 0 then
    t := round(1000/(limit/t))
  else t:=0;
  with legend.date do
    getdate(year,month,date,dayofweek);
end;


procedure getdata(var splot:artype; time:word);

{  Gets data from A/D converter.  }

var i, portnum, code:integer;
begin
  val(getinput('Enter port num: '), portnum, code);
  while (portnum<=0) or (portnum>8) do
    val(getinput('Must be in 1..8:'), portnum, code);
  dec(portnum);
  outtext('Getting data, please wait...');
  for i:=0 to limit do
  begin
    port[board+portnum]:=0;
    delay(time);
    splot[i]:=port[board];
    if graph_driver = hercmono then
    begin
      if i mod 10=0 then
        outtextxy(i div 10,10,chr(22));
    end
    else
    if i mod 12=0 then
      outtextxy(i div 11,10,chr(22));
  end;
  cleardevice;
end;


function rightjustify(astring:string):integer;

{  Used to position title.  }

begin
  rightjustify:=xmax-textwidth(astring)-30;
```

```
end;

procedure axis(legend:legrec);

{   Draws x and y numbered axis with legend printed.  }

var i : integer;

begin
  settextstyle(smallfont,0,4);
  line(30, 0, 30, 296);                                    { vertical axis }
  line(30, 290, 719, 290);                                 { horizontal axis }
  for i := 0 to 130 do
  begin
    if i<=40 then
    begin
      if i mod 4 <> 0 then
        line(round(i*6.25)+30,293, round(i*6.25)+30,290)
      else
        line(round(i*6.25)+30, 296 , round(i*6.25)+30, 290);
    end
    else
    begin
      if i>=50 then
        if i mod 5 <> 0 then
          line(i*5+30, 293, i*5+30, 290)
        else
          line(i*5+30, 296, i*5+30, 290);
    end;

    if i mod 5 = 0 then
      if i<30 then
        outtextxy(i*2*5+20,300,fstr(4000-((i div 5) * 400)))
      else
      if (i>30) and (i<=80) then
        outtextxy((i-1)*2*5-20,300,fstr(2000-(((i-30) div 5) * 200)));

  end;

  if not absorb then
    for i:= 58 downto 0 do
    begin
      if i mod 5 <>3 then
        line(33,i*5,30,i*5)
      else
        line(35,i*5,30,i*5);
      if i mod 10 = 8 then
        outtextxy(30-textwidth(fstr((58-i)*2))-2,i*5-5, fstr((58 - i)*2));
    end
  else
    for i:= 11 downto 0 do
    begin
      if i mod 5 <> 1 then
        line(33,round(i*26.36363636),30,round(i*26.36363636))
      else
      begin
        line(35,round(i*26.36363636),30,round(i*26.36363636));
        outtextxy(20,round(i*26.36363636)-5, fstr((11-i) div 5));
      end;
    end;


  outtextxy(300,320,'Wave Number (CM )');
```

```
    line(389,323,391,323); line(393,321,393,324); putpixel(392,321,1);
    with legend.date do
      outtextxy(5,330,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)+')');
    settextstyle(smallfont,1,4);
    if absorb then
      outtextxy(2,110,'Absorbance')
    else
      outtextxy(2,100,'Percent Transmission');
    settextstyle(0,0,1);
    outtextxy(rightjustify(legend.title),323,legend.title);

  end; { axis }

  procedure axiscga(legend:legrec);

  {   Draws x and y numbered axis with legend printed.  }

  var i : integer;

  begin
    settextstyle(smallfont,0,4);
    line(27, 0, 27, 170);                                    { vertical axis }
    line(27, 167, 639, 167);                                { horizontal axis }
    for i := 0 to 130 do
    begin
      if i<=40 then
      begin
        if i mod 4 <> 0 then
          line(round(i*5.75)+27,168, round(i*5.75)+27,167)
        else
          line(round(i*5.75)+27, 170 , round(i*5.75)+27, 167);
      end
      else
      begin
        if i>=50 then
          if i mod 5 <> 0 then
            line(round(i*4.6)+27, 168, round(i*4.6)+27, 167)
          else
            line(round(i*4.6)+27, 170, round(i*4.6)+27, 167);
      end;

      if i mod 5 = 0 then
        if i<30 then
          outtextxy(round(i*2*4.6)+16,172,fstr(4000-((i div 5) * 400)))
        else
        if (i>30) and (i<=80) then
          outtextxy(round((i-1)*2*4.6)-20,172,fstr(2000-(((i-30) div 5) * 200)));

    end;

    if not absorb then
      for i:= 58 downto 0 do
      begin
        if i mod 5 <>3 then
          line(29,round(i*2.879310345),27,round(i*2.879310345))
        else
          line(31,round(i*2.879310345),27,round(i*2.879310345));
        if i mod 10 = 8 then
          outtextxy(27-textwidth(fstr((58-i)*2))-2,round(i*2.879310345)-5, fstr((5
8 - i)*2));
      end
    else
      for i:= 11 downto 0 do
```

```
    begin
      if i mod 5 <> 1 then
        line(29,round(i*15.18181818),27,round(i*15.18181818))
      else
      begin
        line(31,round(i*15.18181818),27,round(i*15.18181818));
        outtextxy(18,round(i*15.18181818)-5, fstr((11-i) div 5));
      end;
    end;


  outtextxy(267,184,'Wave Number (CM )');
  line(356,186,358,186); line(360,185,360,187); putpixel(359,185,1);
  with legend.date do
    outtextxy(4,190,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)+')');
  if absorb then
  begin
    settextstyle(smallfont,1,4);
    outtextxy(2,50,'Absorbance')
  end
  else
  begin
    settextstyle(smallfont,1,3);
    outtextxy(2,35,'Percent Transmission');
  end;
  settextstyle(0,0,1);
  outtextxy(rightjustify(legend.title),186,legend.title);

end; { axis }


function adjust(a,num:integer):integer;

{  Adjusts digital value to appropriate Y position on screen.  }

begin
  if (a<0) and (num=numberofarrays) then a:=0;
  if graph_driver = hercmono then
    adjust:=round((290/numberofarrays)*num - (279/numberofarrays) * a/255)
  else
    adjust:=round((167/numberofarrays)*num - (160/numberofarrays) * a/255)
end;


procedure addtoarray(add:integer; var plot:plots);

{ Used to move graph up and down. }

var i,num:integer;

begin
  for num:=1 to numberofarrays do
    for i:=0 to limit do
      plot[num][i]:=plot[num][i]+add;
end;


procedure absorbance(var plot:plots);
var num,i:integer;

begin
  for num:=1 to numberofarrays do
    for i := 0 to limit do
```

```
    begin
      if plot[num][i] <= 0 then plot[num][i]:=1;
      plot[num][i]:=round((2-(ln((plot[num][i]/255)*100)/2.302585093))*115);
      if i=0 then moveto(ybord,adjust(plot[num][i],num));
      if i mod xratio = 0 then
        lineto(i div xratio+ybord, adjust(plot[num][i],num));
    end;
end;


procedure draw(var plot:plots);

{  Plots graph then calls integration routine. }

var x,num : integer;

begin
  for num:=1 to numberofarrays do
  begin
    moveto(ybord,adjust(plot[num][0],num));
    for x := 1 to limit do
      if x mod xratio = 0 then
        lineto(x div xratio+ybord, adjust(plot[num][x],num));
  end;
end; { draw }


procedure sorti(xput:st; arlimit:integer; var sindex:st);

{ Used to return an array of sorted indexes according to the passed array.
  It's used to make sure the answers from couple and getratio don't overlap.}

type hold= record
             xp:integer;
             postn:integer;
           end;

var rec: array[0..20] of hold;
    temprec:hold;
    i,j:integer;

begin
  for i:=0 to arlimit do
  begin
    rec[i].xp:=xput[i];
    rec[i].postn:=i;
  end;

  for i:= arlimit downto 1 do
    for j:= 0 to i-1 do
    if rec[j].xp > rec[j+1].xp then
    begin
      temprec:=rec[j];
      rec[j]:=rec[j+1];
      rec[j+1]:=temprec;
    end;

  for i:=0 to arlimit do
    sindex[i]:=rec[i].postn;

end;
```

```
procedure zoom(var plot:plots; legend:legrec);

{  Zoom in on specified area of graph.  }

label redo;
var ch:char;
    x,y,i,xmod,xmod4,sp,num:integer;
    xright,xleft:real;
    shift:byte absolute $40:$17;      { Location of shift status }
    r : real;
    s:string[1];

begin
  xleft:=280;
  xright:=330;
  y:=0;
  putimage(round(xleft),y,aline^,xorput);
  putimage(round(xright),y,aline^,xorput);
  setlinestyle(0,0,2);
  line(round(xleft)+2,0,round(xright),0);

  repeat
    ch:=getkey;
    case ch of
      '<' : if shift and 2 = 2 then
            begin
              if xleft>30 then
              begin
                putimage(round(xleft),y,aline^,xorput);
                if xleft>280 then
                  xleft:=xleft-5
                else
                  xleft:=xleft-6.25;
                line(round(xleft)+2,0,round(xright),0);
                putimage(round(xleft),y,aline^,xorput);
              end;
            end
            else
            if shift and 1 = 1 then
              if xright>xleft+5 then
              begin
                setcolor(0);
                line(round(xleft)+2,0,round(xright),0);
                putimage(round(xright),y,aline^,xorput);
                if xright>280 then
                  xright:=xright-5
                else
                  xright:=xright-6.25;
                setcolor(1);
                line(round(xleft)+2,0,round(xright),0);
                putimage(round(xright),y,aline^,xorput);
              end;


      '>' : if shift and 2 = 2 then
            begin
              if xleft<xright-5 then
              begin
                setcolor(0);
                line(round(xleft)+2,0,round(xright),0);
                putimage(round(xleft),y,aline^,xorput);
                if xleft>=280 then
                  xleft:=xleft+5
```

```
            else
              xleft:=xleft+6.25;
            setcolor(1);
            line(round(xleft)+2,0,round(xright),0);
            putimage(round(xleft),y,aline^,xorput);
          end;
        end
        else
        if shift and 1 = 1 then
          if xright<=700 then
          begin
            putimage(round(xright),y,aline^,xorput);
            if xright>=280 then
              xright:=xright+5
            else
              xright:=xright+6.25;
            line(round(xleft)+2,0,round(xright),0);
            putimage(round(xright),y,aline^,xorput);
          end;


ctrlleft : if xleft>30 then
           begin
             setcolor(0);
             line(round(xleft)+2,0,round(xright),0);
             putimage(round(xleft),y,aline^,xorput);
             putimage(round(xright),y,aline^,xorput);
             if xleft>280 then
               xleft:=xleft-5
             else
               xleft:=xleft-6.25;
             if xright>280 then
               xright:=xright-5
             else
               xright:=xright-6.25;
             setcolor(1);
             line(round(xleft)+2,0,round(xright),0);
             putimage(round(xleft),y,aline^,xorput);
             putimage(round(xright),y,aline^,xorput);
           end;


ctrlright : if xright<=700 then
            begin
              setcolor(0);
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xleft),y,aline^,xorput);
              putimage(round(xright),y,aline^,xorput);
              if xleft>=280 then
                xleft:=xleft+5
              else
                xleft:=xleft+6.25;
              if xright>=280 then
                xright:=xright+5
              else
                xright:=xright+6.25;
              setcolor(1);
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xleft),y,aline^,xorput);
              putimage(round(xright),y,aline^,xorput);
            end;
```

```
      return : begin
                 xleft:=xleft-30;
                 xright:=xright-30;
                 cleardevice;                          {              }
Redo :           settextstyle(smallfont, 0, 4);        {  Draw Axis   }
                 setlinestyle(0,0,1);                  {              }
                 line(30, 0, 30, 296);
                 line(30, 290, 719, 290);
                 sp:=round(650/((xright-xleft)/5));
                 xmod:=5-round(frac(xleft/25) * 5);
                 xmod4:=4-round(frac(xleft/25) * 4);
                 if xmod=5 then xmod:=0;
                 if xmod4=4 then xmod4:=0;

                 for i := 0 to round((xright-xleft)/5) do
                 begin
                   if round(xleft+i*6.25) < 250 then
                   begin
                     if i mod 4 <> xmod4 then
                       line(round(i*sp*1.25+30), 293, round(i*sp*1.25+30), 290)
                     else
                     begin
                       line(round(i*sp*1.25+30), 296, round(i*sp*1.25+30), 290);
                       outtextxy(round(i*sp*1.25+20), 300, fstr(round(36-(((xlef
t-55+i*6.25)/25)*2))*100));
                     end;
                   end
                   else
                   if round(xleft)+i*5 >= 249-frac(xleft)/0.25 then
                   begin
                     if i mod 5 <> xmod then
                       line(i*sp+30,293, i*sp+30,290)
                     else
                     begin
                       line(i*sp+30, 296, i*sp+30, 290);
                       outtextxy(i*sp+20,300,fstr((27-((round(xleft)-55+i*5) div
 25))*100));
                     end;
                   end;
                 end;

                 if not absorb then
                   for i:= 58 downto 0 do
                   begin
                     if i mod 5 <>3 then
                       line(33,i*5,30,i*5)
                     else
                       line(35,i*5,30,i*5);
                     if i mod 10 = 8 then
                     outtextxy(30-textwidth(fstr((58-i)*2))-2,i*5-5, fstr((58 -
i)*2));
                   end
                 else
                   for i:= 11 downto 0 do
                   begin
                     if i mod 5 <> 1 then
                       line(33,round(i*26.36363636),30,round(i*26.36363636))
                     else
                     begin
                       line(35,round(i*26.36363636),30,round(i*26.36363636));
                       outtextxy(20,round(i*26.36363636)-5, fstr((11-i) div 5));
                     end;
                   end;
```

```
                    outtextxy(300,320,'Wave Number (CM )');
                    line(389,323,391,323); line(393,321,393,324); putpixel(392,321,
1);
                    with legend.date do
                      outtextxy(5,330,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)
+')');
                    settextstyle(smallfont,1,4);
                    if absorb then
                      outtextxy(2,110,'Absorbance')
                    else
                      outtextxy(2,100,'Percent Transmission');
                    settextstyle(0,0,1);
                    outtextxy(rightjustify(legend.title),323,legend.title);

                    r:=xratio*(xright-xleft)/650;          {  Plot Zoom  }
                    for num:=1 to numberofarrays do
                    begin
                      moveto(30,adjust(plot[num][round(xleft)*xratio],num));
                      for x:=1 to 650 do
                      if round(x*r+xleft*xratio) <= limit then
                        lineto(x+30,adjust(plot[num][round(x*r+xleft*xratio)],num))
;
                    end;

                    repeat
                      ch:=getkey;
                      case ch of
                                                        {  for saving and     }
                        F2 : scrsave;                   {  restoring screens. }

                        F3 : scrrestore;

                      end;
                    until (ch=esc);
                    cleardevice;
                    axis(legend);
                    draw(plot);
                  end;

        esc : begin
                setcolor(0);
                line(round(xleft)+2,0,round(xright),0);
                putimage(round(xleft),y,aline^,xorput);
                putimage(round(xright),y,aline^,xorput);
                setcolor(1);
                setlinestyle(0,0,normwidth);
              end;
      end;

    until (ch=esc);

end;


procedure compare(var plot:plots; var legend:legrec);
var err : boolean;
    ot:string;

begin
  if numberofarrays<3 then
  begin
    ot:=legend.title;
```

```
    readdata(plot[numberofarrays+1],legend,err,'File to compare:');
    if not err then
    begin
      cleardevice;
      inc(numberofarrays);
      oldtitle[numberofarrays]:=legend.title;
      legend.title:=ot+'+'+legend.title;
      if graph_driver=hercmono then
        axis(legend)
      else
        axiscga(legend);
      draw(plot);
    end;
  end;
end;

procedure remove(var plot:plots; var legend:legrec);
begin
  if numberofarrays>1 then
  begin
    with legend do
      delete(title, pos('+'+oldtitle[numberofarrays],title), length(oldtitle[num
berofarrays])+1);
    cleardevice;
    dec(numberofarrays);
    if graph_driver=hercmono then
      axis(legend)
    else
      axiscga(legend);
    draw(plot);
  end;
end;


procedure movegr(plot:plots; legend:legrec);

{  Main procedure to process graph. }

var ch:char;

begin
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  addtoarray(5,plot);
                  cleardevice;
                  axis(legend);
                  draw(plot);
                end;

      downarrow : begin
                    addtoarray(-5,plot);
                    cleardevice;
                    axis(legend);
                    draw(plot);
                  end;

      'z','Z' : zoom(plot, legend);

      'a','A' : begin
                  absorb:=true;
                  cleardevice;
```

```
                    axis(legend);
                    absorbance(plot);
                  end;

        'c','C' : compare(plot,legend);

        'r','R' : remove(plot,legend);

        F2 : scrsave;

        F3 : scrrestore;

        altM : ;

      end;
    until ch=#27;
  end;


  procedure initcga;
  var size:integer;
  begin
    bar(178,56,285,63);
    size:=imagesize(178,56,285,63);
    getmem(sbar,size);
    getimage(178,56,285,63,sbar^);
    cleardevice;
    moveto(89,14);
    lineto(102,12);
    lineto(92,8);
    lineto(89,14);
    floodfill(93,12,1);
    setlinestyle(0,0,thickwidth);
    line(94,13,101,21);
    line(96,13,103,21);
    size:=imagesize(89,8,104,22);
    getmem(arrow,size);
    getimage(89,8,104,22,arrow^);
    setlinestyle(0,0,normwidth);
    cleardevice;
    line(31,0,31,167);
    size:=imagesize(30,0,32,167);
    getmem(aline,size);
    getimage(30,0,32,167,aline^);
    cleardevice;
    size:=imagesize(58,23,294,86);
    getmem(blank,size);
    getimage(58,23,294,86,blank^);

  end;


  procedure selectcga(var choice:chtype);
  var x,y:integer;
      ch:char;

  begin
    x:=256; y:=93;
    cleardevice;
    settextstyle(gothicfont,horizdir,7);
    outtextxy(203,0,'Infrared');
    rectangle(243,82,376,169);
    settextstyle(smallfont,0,4);
```

```
  outtextxy(261,91,'A/D Converter');
  outtextxy(261,100,'Load IR');
  outtextxy(261,109,'Save IR');
  outtextxy(261,118,'Load Ascii');
  outtextxy(261,127,'Save Ascii');
  outtextxy(261,136,'Review Graph');
  outtextxy(261,145,'OS shell');
  outtextxy(261,154,'Quit');
  putimage(x,y,sbar^,xorput);
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  putimage(x,y,sbar^,xorput);
                  if y=93 then y:=156
                  else y:=y-9;
                  putimage(x,y,sbar^,xorput);
                end;

      downarrow : begin
                    putimage(x,y,sbar^,xorput);
                    if y=156 then y:=93
                    else y:=y+9;
                    putimage(x,y,sbar^,xorput);
                  end;

      return : case y of
                 93 : choice:=ADconvert;

                 102 : choice:=loadIR;

                 111 : choice:=saveIR;

                 120 : choice:=loadasc;

                 129 : choice:=saveasc;

                 138 : choice:=review;

                 147 : choice:=OSshell;

                 156 : choice:=quit;
               end;

    end;
  until ch=return;

end;  { selectcga }


procedure zoomcga(var plot:plots; legend:legrec);

{  Zoom in on specified area of graph.  }

label redo;
var ch:char;
    x,y,i,xmod,xmod4,sp,num:integer;
    xright,xleft:real;
    shift:byte absolute $40:$17;      { Location of shift status }
    r : real;
    s:string[1];

begin
```

```
xleft:=257;
xright:=303;
y:=0;
putimage(round(xleft),y,aline^,xorput);
putimage(round(xright),y,aline^,xorput);
setlinestyle(0,0,2);
line(round(xleft)+2,0,round(xright),0);

repeat
  ch:=getkey;
  case ch of
    '<' : if shift and 2 = 2 then
          begin
            if xleft>27 then
            begin
              putimage(round(xleft),y,aline^,xorput);
              if xleft>257 then
                xleft:=xleft-4.6
              else
                xleft:=xleft-5.75;
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xleft),y,aline^,xorput);
            end;
          end
          else
          if shift and 1 = 1 then
            if xright>xleft+4.6 then
            begin
              setcolor(0);
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xright),y,aline^,xorput);
              if xright>257 then
                xright:=xright-4.6
              else
                xright:=xright-5.75;
              setcolor(1);
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xright),y,aline^,xorput);
            end;


    '>' : if shift and 2 = 2 then
          begin
            if xleft<xright-4.6 then
             begin
               setcolor(0);
               line(round(xleft)+2,0,round(xright),0);
               putimage(round(xleft),y,aline^,xorput);
               if xleft>=257 then
                 xleft:=xleft+4.6
               else
                 xleft:=xleft+5.75;
               setcolor(1);
               line(round(xleft)+2,0,round(xright),0);
               putimage(round(xleft),y,aline^,xorput);
             end;
           end
           else
           if shift and 1 = 1 then
             if xright<=630 then
              begin
                putimage(round(xright),y,aline^,xorput);
                if xright>=257 then
```

```
                xright:=xright+4.6
              else
                xright:=xright+5.75;
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xright),y,aline^,xorput);
            end;


      ctrlleft : if xleft>27 then
                 begin
                   setcolor(0);
                   line(round(xleft)+2,0,round(xright),0);
                   putimage(round(xleft),y,aline^,xorput);
                   putimage(round(xright),y,aline^,xorput);
                   if xleft>257 then
                     xleft:=xleft-4.6
                   else
                     xleft:=xleft-5.75;
                   if xright>257 then
                     xright:=xright-4.6
                   else
                     xright:=xright-5.75;
                   setcolor(1);
                   line(round(xleft)+2,0,round(xright),0);
                   putimage(round(xleft),y,aline^,xorput);
                   putimage(round(xright),y,aline^,xorput);
                 end;


      ctrlright : if xright<=630 then
                  begin
                    setcolor(0);
                    line(round(xleft)+2,0,round(xright),0);
                    putimage(round(xleft),y,aline^,xorput);
                    putimage(round(xright),y,aline^,xorput);
                    if xleft>=257 then
                      xleft:=xleft+4.6
                    else
                      xleft:=xleft+5.75;
                    if xright>=257 then
                      xright:=xright+4.6
                    else
                      xright:=xright+5.75;
                    setcolor(1);
                    line(round(xleft)+2,0,round(xright),0);
                    putimage(round(xleft),y,aline^,xorput);
                    putimage(round(xright),y,aline^,xorput);
                  end;


      return : begin
                 xleft:=xleft-27;
                 xright:=xright-27;
                 cleardevice;                          {              }
Redo :           settextstyle(smallfont, 0, 4);        {  Draw Axis   }
                 setlinestyle(0,0,1);                  {              }
                 line(27, 0, 27, 170);
                 line(27, 167, 639, 167);
                 sp:=round(598/((xright-xleft)/4.6));
                 xmod:=5-round(frac(xleft/23) * 5);
                 xmod4:=4-round(frac(xleft/23) * 4);
                 if xmod=5 then xmod:=0;
                 if xmod4=4 then xmod4:=0;
```

```
                    for i := 0 to round((xright-xleft)/4.6) do
                    begin
                      if round(xleft+i*5.75) < 230 then
                      begin
                        if i mod 4 <> xmod4 then
                          line(round(i*sp*1.25+27), 168, round(i*sp*1.25+27), 167)
                        else
                        begin
                          line(round(i*sp*1.25+27), 170, round(i*sp*1.25+27), 167);
                          outtextxy(round(i*sp*1.25+16), 172, fstr(round(36-(((xlef
t-50+i*5.75)/23)*2))*100));
                        end;
                      end
                      else
                      if round(xleft)+i*4.6 >= 226+frac(xleft)/0.25 then
                      begin
                        if i mod 5 <> xmod then
                          line(i*sp+27,168, i*sp+27,167)
                        else
                        begin
                          line(i*sp+27, 170, i*sp+27, 167);
                          outtextxy(i*sp+17,172,fstr(round(28-((xleft-50+i*4.6)/23)
)*100));
                        end;
                      end;
                    end;

                    if not absorb then
                      for i:= 58 downto 0 do
                      begin
                        if i mod 5 <>3 then
                          line(29,round(i*2.879310345),27,round(i*2.879310345))
                        else
                          line(31,round(i*2.879310345),27,round(i*2.879310345));
                        if i mod 10 = 8 then
                          outtextxy(27-textwidth(fstr((58-i)*2))-2,round(i*2.879310
345)-5, fstr((58 - i)*2));
                      end
                    else
                      for i:= 11 downto 0 do
                      begin
                        if i mod 5 <> 1 then
                          line(29,round(i*15.18181818),27,round(i*15.18181818))
                        else
                        begin
                          line(31,round(i*15.18181818),27,round(i*15.18181818));
                          outtextxy(18,round(i*15.18181818)-5, fstr((11-i) div 5));
                        end;
                      end;

                    outtextxy(267,184,'Wave Number (CM )');
                    line(356,186,358,186); line(360,185,360,187); putpixel(359,185,
1);

                    with legend.date do
                    outtextxy(4,190,'('+fstr(month)+'/'+fstr(date)+'/'+fstr(year)+'
)');

                    if absorb then
                    begin
                      settextstyle(smallfont,1,4);
                      outtextxy(2,50,'Absorbance')
                    end
                    else
```

```
                begin
                  settextstyle(smallfont,1,3);
                  outtextxy(2,35,'Percent Transmission');
                end;
                settextstyle(0,0,1);
                outtextxy(rightjustify(legend.title),186,legend.title);

                r:=xratio*(xright-xleft)/598;          {  Plot Zoom  }
                for num:=1 to numberofarrays do
                begin
                  moveto(27,adjust(plot[num][round(xleft)*xratio],num));
                  for x:=1 to 598 do
                  if round(x*r+xleft*xratio) <= limit then
                    lineto(x+27,adjust(plot[num][round(x*r+xleft*xratio)],num))
;
                end;

                repeat
                  ch:=getkey;
                  case ch of
                                                    {  for saving and     }
                    F2 : scrsavecga;                {  restoring screens. }

                    F3 : scrrestorecga;

                  end;
                until (ch=esc);
                cleardevice;
                axiscga(legend);
                draw(plot);
              end;

      esc : begin
              setcolor(0);
              line(round(xleft)+2,0,round(xright),0);
              putimage(round(xleft),y,aline^,xorput);
              putimage(round(xright),y,aline^,xorput);
              setcolor(1);
              setlinestyle(0,0,normwidth);
            end;
    end;

  until (ch=esc);

end;


procedure movegrcga(plot:plots; legend:legrec);
var ch:char;

begin
  repeat
    ch:=getkey;
    case ch of
      uparrow : begin
                  addtoarray(5,plot);
                  cleardevice;
                  axiscga(legend);
                  draw(plot);
                end;

      downarrow : begin
                    addtoarray(-5,plot);
```

```
                    cleardevice;
                    axiscga(legend);
                    draw(plot);
                  end;

      'a','A' : begin
                    absorb:=true;
                    cleardevice;
                    axiscga(legend);
                    absorbance(plot);
                  end;

      'z','Z' : zoomcga(plot, legend);

      'c','C' : compare(plot, legend);

      'r','R' : remove(plot,legend);

      F2 : scrsavecga;

      F3 : scrrestorecga;

      altM : ;
    end;
  until ch=#27;
end;


{ ********************************************************************************
  *                                Main Block                                    *
  ********************************************************************************}

begin

  initialize;

  if graph_driver = hercmono then
  begin
    initherc;

    repeat
      xratio:=10; yratio:=255;
      absorb:=false;
      numberofarrays:=1;
      select(choice);                                  { select input source }
      case choice of
        ADconvert : begin
                      cleardevice;
                      getuserinp(time,legend);
                      getdata(plot[1],time);
                      axis(legend);
                      draw(plot);
                      movegr(plot,legend);
                    end;

        loadIR : begin
                       cleardevice;
                       readdata(plot[1],legend,error,'File to read:');
                       if not error then
                       begin
                         cleardevice;
                         axis(legend);
                         draw(plot);
```

```
                    movegr(plot,legend);
                  end;
                end;

      saveIR : savedata(plot[1],legend);

      loadasc : begin
                  cleardevice;
                  loadascii(plot[1],error);
                  if not error then
                  begin
                    cleardevice;
                    legend.title:=getinput('Enter graph title:');
                    with legend.date do
                      getdate(year,month,date,dayofweek);
                    axis(legend);
                    draw(plot);
                    movegr(plot,legend);
                  end;
                end;

      saveasc : saveascii(plot[1]);

      review : begin
                 cleardevice;
                 axis(legend);
                 draw(plot);
                 movegr(plot,legend);
               end;

      OSshell : begin
                  restorecrtmode;
                  writeln('Type EXIT to return to menu...');
                  swapvectors;
                  exec(getenv('comspec'),'/c '+getenv('comspec'));
                  swapvectors;
                  setgraphmode(graph_mode);
                end;

    end;
  until(choice = quit)
end
else
begin

  initcga;

  repeat
    xratio:=11; yratio:=255;
    numberofarrays:=1;
    absorb:=false;
    selectcga(choice);
    case choice of
      ADconvert : begin
                    cleardevice;
                    getuserinp(time,legend);
                    getdata(plot[1],time);
                    axiscga(legend);
                    draw(plot);
                    movegrcga(plot,legend);
                  end;

      loadIR : begin
```

```
                    cleardevice;
                    readdata(plot[1],legend,error,'File to read:');
                    if not error then
                    begin
                      cleardevice;
                      axiscga(legend);
                      draw(plot);
                      movegrcga(plot,legend);
                    end;
                  end;

        saveIR : savedata(plot[1],legend);

        loadasc : begin
                    cleardevice;
                    loadascii(plot[1],error);
                    if not error then
                    begin
                      legend.title:=getinput('Enter graph title:');
                      with legend.date do
                        getdate(year,month,date,dayofweek);
                      axiscga(legend);
                      draw(plot);
                      movegrcga(plot,legend);
                    end;
                  end;

        saveasc : saveascii(plot[1]);

        review : begin
                   cleardevice;
                   axiscga(legend);
                   draw(plot);
                   movegrcga(plot,legend);
                 end;

        OSshell : begin
                    restorecrtmode;
                    writeln('Type EXIT to return to menu...');
                    swapvectors;
                    exec(getenv('comspec'),'/c '+getenv('comspec'));
                    swapvectors;
                    setgraphmode(graph_mode);
                  end;

      end;

    until(choice = quit);

  end;

  closegraph;

end. { main }
```

Spectronic 20D

Interfacing the Spectronic 20D

The Spectronic 20D has provision for interfacing a computer via an RS-232 serial connection. This feature was used, along with a voice synthesiser, to provide spoken output of the wavelength and absorption or transmission. It is hoped that this might enable a visually handicapped student to participate in experiments involving the Spectronic 20.

The equipment was very simple; a 3-wire connection between the Spectronic 20D and a serial port on the computer (see Appendix 3) and "The Speech Thing" by Covox Inc. connected in series with the computer's printer cable. No cards or connections are needed inside the computer. The quality of the synthesised speech was adequate for the purpose.

```
uses dos,crt,graph,drivers;

{ This program is used to interface the computer with the Spectronic 20D.
  The program, using "The Speech Thing", can say outloud values from the
  Spectronic 20. It can display the data graphically and save it to a text
  file for later printing. You may modify the program as you wish.

                                          Colin Lamarre
                                          August, 1989
                                                      }

const null=#0;
      return=#13;
      esc=#27;
      uparrow=#200;
      downarrow=#208;
      leftarrow=#203;
      rightarrow=#205;

type datatype=record
                wavelength:integer;
                data:real;
                mode:char;
              end;

     ardata=array[1..1000] of datatype;

var regs:registers;
    send,ch:char;
    gd,gm,i:integer;
    instr:string;
    indata:datatype;
    data1000:ardata;
    st:text;
    blank:pointer;

function GetKey:char;
var
  C : Char;
begin
  C := ReadKey;
  repeat
    if C = NULL then
    begin
      C := ReadKey;
      if Ord(C) > 127 then
        C := NULL
      else
        GetKey := Chr(Ord(C) + 128);
    end
    else
      GetKey := C;
  until C <> NULL;
end; { GetKey }
```

```
function getinput(message:string):string;

{* Lets user type input in graphics mode. It saves the screen section
   then replaces it when done.                                        *}

var ch : char;
    cursor,csize,y : integer;
    t : string;
    replace:pointer;

begin
  getmem(replace,imagesize(65,40,330,150));
  getimage(65,40,330,150,replace^);
  putimage(65,40,blank^,normalput);
  settextstyle(smallfont,0,7);
  csize := textwidth('A');
  t:='';
  cursor:=80; y:=77;
  outtextxy(75,50,message);
  rectangle(75,75,300,100);
  line(cursor,96,cursor+csize,96);
  repeat
    ch:=getkey;
    case ch of
      #8,leftarrow : begin
                       if cursor >= 80+csize then
                       begin
                         setcolor(0);
                         line(cursor,96,cursor+csize,96);
                         cursor:=cursor-csize;
                         outtextxy(cursor,y,t[length(t)]);
                         setcolor(1);
                         line(cursor,96,cursor+csize,96);
                         delete(t,length(t),1);
                       end;
                     end;

      rightarrow, uparrow, downarrow, #9, #186..#196::


      return : begin
                 getinput:=t;
                 putimage(65,40,replace^,normalput);
               end;

      Esc : begin
              getinput:='';
              putimage(65,40,replace^,normalput);
            end;


      else  begin
              if cursor  = 80+csize*16 then
              begin
```

```
            t:=t+ch;
            setcolor(0);
            line(cursor,96,cursor+csize,96);
            setcolor(1);
            outtextxy(cursor,y,ch);
            cursor:=cursor+csize;
            line(cursor,96,cursor+csize,96);
          end;
        end;

    end;
  until (ch=return) or (ch=esc);
end;

procedure mainmenu(var ch:char);
begin
  cleardevice;
  outtextxy(15,1,'   What would you like to do?');
  outtextxy(15,25,'   1 - Receive data');
  outtextxy(15,40,'   2 - Set the data mode to absorbance');
  outtextxy(15,55,'   3 - Set the data mode to transmittance');
  outtextxy(15,70,'   4 - Set the data mode to concentration');
  outtextxy(15,85,'   5 - Plot');
  outtextxy(15,100,'   6 - speak vals');
  outtextxy(15,115,'   7 - Quit');
  outtextxy(15,140,'   Enter your choice: ');
  repeat
    ch:=getkey;
    case ch of
      '1' : ch:='P';
      '2' : ch:='A';
      '3' : ch:='T';
      '4' : ch:='C';
      '5' : ch:='5';
      '6' : ch:='6';
      '7' : ch:='Q';
     else begin
            ch:=#0;
            sound(500);
            delay(50);
            nosound;
          end;
    end;
  until ch in ['P','A','T','C','5','6','Q'];

end;


procedure wait;
var ch:char;

begin
  outtextxy(1,50,'Done -- press any key to continue');
  repeat until keypressed;
  ch:=readkey;
```

```
end;


procedure initcom;    { set baud rate, etc.. }
var size:integer;
begin
  cleardevice;
  size:=imagesize(65,40,330,150);
  getmem(blank,size);
  getimage(65,40,330,150,blank^);
  regs.ah:=0;
  regs.dx:=0;
  regs.al:=$87;    { 10000011 }
  intr($14,regs);
end;

function comreadv:boolean;
begin
  regs.ah:=3;
  regs.dx:=0;
  intr($14,regs);
  if regs.ah and 1 = 1 then
    comreadv:=true
  else
    comreadv:=false;
end;

function getcomchar:char;

begin
  regs.ah:=2;
  regs.dx:=0;
  intr($14,regs);
  if regs.ah=0 then
    getcomchar:=chr(regs.al);
end;

procedure sendchar(ch:char);

begin
  regs.ah:=1;
  regs.dx:=0;
  regs.al:=ord(ch);
  intr($14,regs);
end;

procedure getdata(var s:string);   { Gets data from the Spec. 20 }
begin
  sendchar('P');
  repeat
    ch:=getcomchar;
  until ch in ['0'..'9'];
  s:=ch;
  repeat
    s:=s+getcomchar;
```

```
    until s[length(s)] in ['A','T','C','F'];
  end;

  procedure drawaxis;
  begin
    line(30,1,30,300);
    line(30,290,719,290);
    outtextxy(25,315,'(Return) to read   (V)iew data   (Esc) when finished   (S)ave to a file');
  end;


  procedure change(instr:string; var indata:datatype);   { Parse data }
  var s:string;
      i,code:integer;

  begin
    s:='';
    i:=1;
    while instr[i] in ['0'..'9'] do
    begin
      s:=s+instr[i];
      inc(i);
    end;

    val(s,indata.wavelength,code);

    s:='';
    repeat
      inc(i);
    until instr[i] in ['0'..'9','-','.'];

    while instr[i] in ['0'..'9','-','.'] do
    begin
      s:=s+instr[i];
      inc(i);
    end;

    val(s,indata.data,code);

    indata.mode:=instr[length(instr)];

  end;


function adjustwave(j:integer):integer;
begin
  adjustwave:=round(699*((j-311)/698))+30;
end;

function adjust(x:real;mode:char):integer;
begin
  case mode of
    'T' : begin   { Transmittance }
            if x=1999.0 then x:=199.9;
            adjust:=290-round(279*(x/200));
```

```
        end;

    'A' : begin   { Absorbance }
            if x=1999.0 then x:=2.0;
            adjust:=290-round(279*(x/2));
          end;

    'C' : begin   { Concentration }
            if x=1999.0 then x:=4.0;
            adjust:=290-round(279*(x/4));
          end;
  end;
end;

function savmode(mode:char):string;
begin
  case mode of
    'A' : savmode:='Absorbance';
    'T' : savmode:='Transmittance';
    'C' : savmode:='Concentration';
    'F' : savmode:='Factor';
  end;
end;

function gettalkd(cdata:datatype;t:boolean):string;
var talkd:string;
begin
  case cdata.mode of
    'T' : str(cdata.data:5:1,talkd);
    'A' : str(cdata.data:5:3,talkd);
    'C','F' : str(cdata.data:5:2,talkd);
  end;
  if t then
  begin
    insert(' point ', talkd, pos('.',talkd));
    delete(talkd,pos('.',talkd),1);
  end;
  gettalkd:=talkd;
end;

procedure save(var data100:ardata; count:integer);

{ Save the data to a printable file. }

var f:text;
    name:string;
    i:integer;

begin
  name:=getinput('Enter file name:');
  if name<>'' then
  begin
    assign(f,name);
    rewrite(f);
    writeln(f,' Wavelength       Data        Mode ');
```

```
    writeln(f);
    writeln(f,'----------------------------------------');
    writeln(f);
    for i:=1 to count do
      writeln(f,' '.data1000[i].wavelength,'            '.gettalkd(data1000[i],false),'       ',
              savmode(data1000[i].mode));
    close(f);
  end;
end;


procedure doplot(var data1000:ardata);
var cdata:datatype;
    sdata,talkw,s:string;
    i,j,v,count:integer;
    x:real;
    ch,c:char;
    oldwave:integer;

begin
  cleardevice;
  outtext('Adjust your dial and press a key to begin.');
  repeat until keypressed;
  ch:=readkey;
  cleardevice;
  i:=1;
  drawaxis;
  getdata(sdata);
  change(sdata,cdata);
  if cdata.mode <> 'F' then
    moveto(adjustwave(cdata.wavelength), adjust(cdata.data,cdata.mode));
  str(cdata.wavelength,talkw);
  write(st,'starting at wavylength equals '+talkw+'    '+savmode(cdata.mode)+' equals '+gettalkd(cdata,true)+'.');
  oldwave:=cdata.wavelength;
  data1000[i]:=cdata;
  inc(i);

  repeat
    ch:=getkey;
    if ch=return then
    begin
      getdata(sdata);
      change(sdata,cdata);
      if cdata.wavelength > oldwave then
      begin
        oldwave:=cdata.wavelength;
        if cdata.mode <> 'F' then
          lineto(adjustwave(cdata.wavelength), adjust(cdata.data,cdata.mode));
        str(cdata.wavelength,talkw);
        write(st,'wavylength equals '+talkw+'    '+savmode(cdata.mode)+' equals '+gettalkd(cdata,true)+'.');
        data1000[i]:=cdata;
        inc(i);
      end;
    end
    else
```

```
    if (ch='s') or (ch='S') then
      save(data1000,i-1)
    else
    if (ch='v') or (ch='V') then
    begin
      for count:=0 to (i-1) div 20 do
      begin
        cleardevice;
        outtextxy(2,1,' Wavelength        Data         Mode ');
        outtextxy(2,21,'---------------------------------------');
        for j:=count*20+1 to (count+1)*20 do
        if j<=i-1 then
        begin
          str(data1000[j].wavelength,s);
          outtextxy(2,(j-count*20)*10+21,'  '+s+'            '+gettalkd(data1000[j],false)+'       '+
                    savmode(data1000[j].mode));
        end;
        outtextxy(2,300,'Press a key for more...');
        repeat until keypressed;
        c:=readkey;
      end;
      cleardevice;
      drawaxis;
      moveto(adjustwave(data1000[1].wavelength),adjust(data1000[1].data,data1000[1].mode));
      for j:=2 to i-1 do
        lineto(adjustwave(data1000[j].wavelength),adjust(data1000[j].data,data1000[j].mode));
    end;
  until ch=esc;
  sound(500);
  delay(50);
  nosound;
end;

procedure constspeak;
var cdata:datatype;
    sdata,talkw:string;
    ch:char;

begin
  cleardevice;
  outtext('Press (esc) to stop.');
  ch:=#0;
  repeat
    getdata(sdata);
    change(sdata,cdata);
    str(cdata.wavelength,talkw);
    write(st,'wavvlength equals '+talkw+'    '+savmode(cdata.mode)+' equals '+gettalkd(cdata,true)+'.');
    delay(5000);
    if keypressed then
      ch:=getkey;
  until ch=esc;
  sound(500);
  delay(50);
  nosound;
end;
```

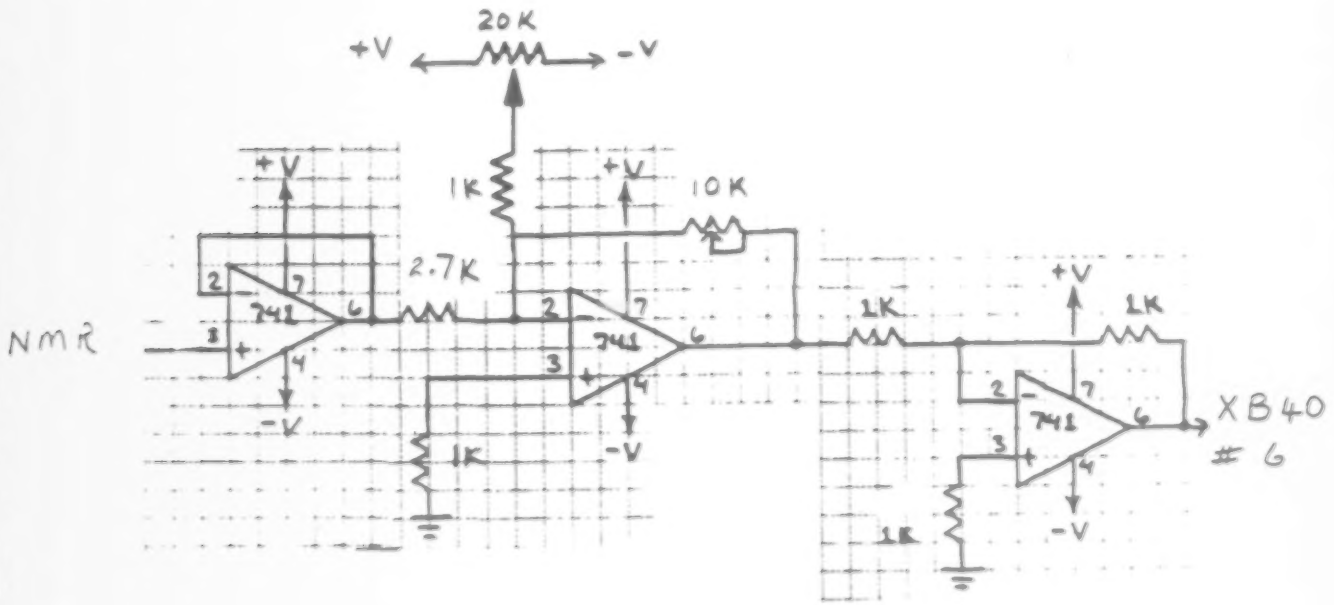
```
begin
  assign(st,'st');
  rewrite(st);
  if registerbgidriver(@hercdriverproc) < 0 then writeln('error');
  detectgraph(gd,gm);
  initgraph(gd,gm,'');
  initcom;
  repeat
    mainmenu(send);
    case send of
      '0' : begin
              cleardevice;
              getdata(instr);
              outtextxy(1,1,instr);
              change(instr,indata);
              wait;
            end;

      'A','T','C' : begin
                      cleardevice;
                      sendchar(send);
                    end;

      'S' : doplot(data1000);

      'b' : constspeak;

    end;
  until send='Q';
  closegraph;
end.
```

APPENDIX 1

ADA300

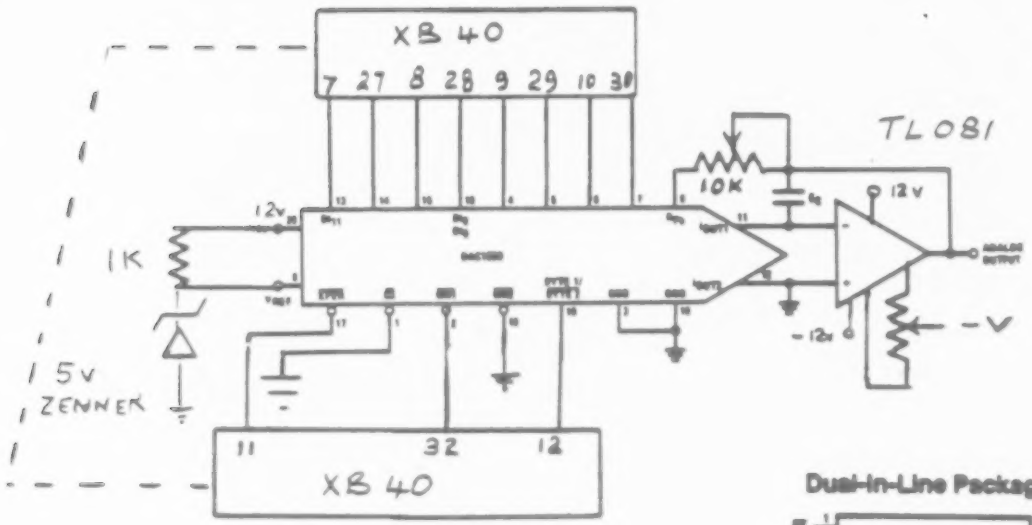
The ADA300 interface card by Real Time Devices Inc. contains an 8 channel 8-bit analog to digital converter (ADC) with 0 to +5V input, a single 8-bit digital to analog converter (DAC) which produces 0 to + 2.56 or 0 to 10V output and 24 TTL digital input/output lines. The 8-bit ADCs convert input voltages between 0 and + 5 volts into digital values between 0 and 255. One of the inputs was used to digitise the output of the NMR receiver which is available as an output for use with the auxiliary oscilloscope. Since this output is much less than 5 volts it was amplified by the operational amplifier circuit shown below. This circuit serves to adjust the input to the ADC so that it swings between 0 and + 5 volts when the EM-360 is operating normally. Whenever the computer is to be used with the NMR the same normal adjustments will have to be made otherwise the operational amplifier controls will have to be altered to ensure that the input to the ADC is in the range 0 to +5V.

If the computer is to be used only to collect data the sweep duration should be set at 2 minutes and the NMR sweep commenced at a delta of 10. The computer should then collect data down to a delta of 0 and display all signals at their correct shifts.



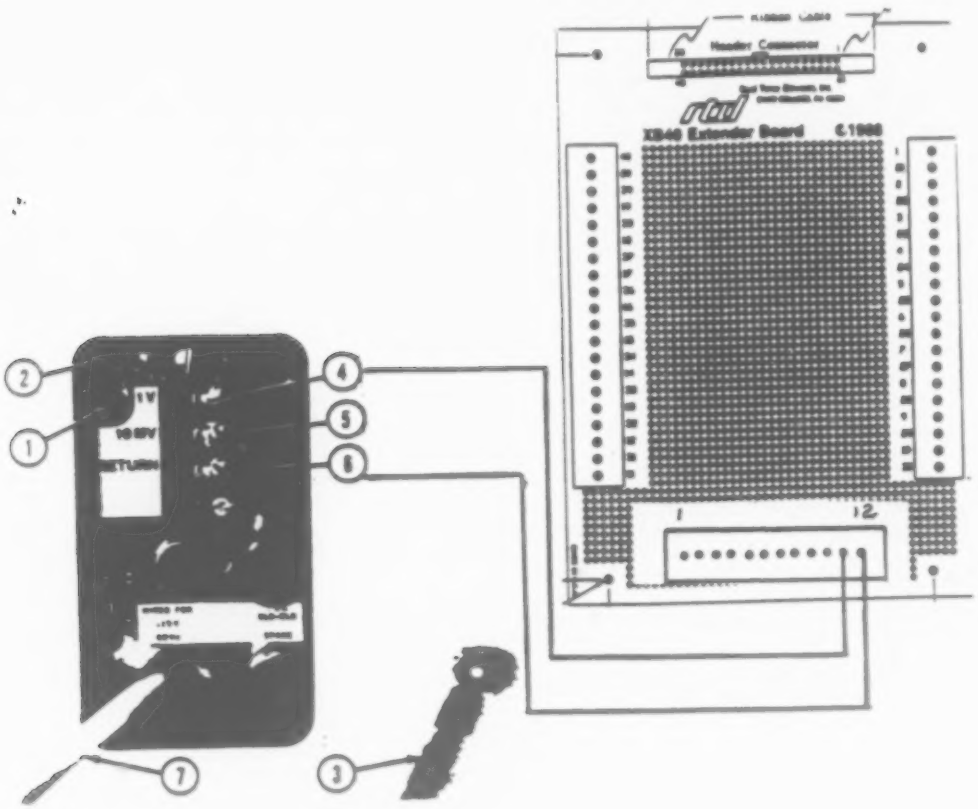
Appendix1 continued.

The circuit shown below was used to provide a sweep voltage between 0 and 5 volts to control the NMR field. The XB40 is the prototyping board which connects to the ADA300.



APPENDIX 2

The connections between the 1 volt output from the IR spectrophotometer and the ADA300 prototyping board are shown in the following diagram. The input amplifier was adjusted to produce an output of 5 volts to the ADA300 board when the output from the IR was 1 volt. It is important to make sure that the zero and 1 volt output controls on the IR have been adjusted as described in the operating manual.



1 *10 mV Readout Adjustment*
2 *1 Volt Readout Adjustment*
3 *Grounding Strap*
4 *1 Volt Recorder Terminal*
5 *10 mV Recorder Terminal*
6 *RETURN Terminal*
7 *Line Power Cord*

APPENDIX 3

The connections between the digital Spectronics 20 and the BEST AT computer are shown in the following diagram. Data can be obtained over the RS-232 connection using any terminal program such as PROCOMM or ASCII Express. Each time a P is sent to the Spectronics 20 it will respond with the current data which is displayed on the digital display. Though the interface will operate at higher baud rates it appears that it is not possible to change the baud rate without first switching off the Spectronics 20 and then switching it on again. It seems best to always operate at 1200 baud which is the default value. Note that commands must be in upper case.

Jones (flat pin connector) at Spectronics 20. Pin #		DB-25S (female) at computer. Pin #
2/3	----------	7
4	----------	3
5	----------	2